

树莓派开发从零开始学

超好玩的智能小硬件制作书



- 树莓派（Raspberry Pi）是近年来最成功的迷你电脑项目
- 全面介绍树莓派的安装和配置、开发环境、开发语言、开发方法和技巧
- 详解移动小车、无人机、报警器、闪烁报警灯等智能小硬件制作实例

胡松涛 编著


本书示例项目源代码

清华大学出版社

树莓派开发从零开始学

超好玩的智能小硬件制作书

胡松涛 编著



清華大学出版社
北 京

内 容 简 介

本书以实战开发为出发点，以 Raspberry Pi 应用开发为主线，通过 Python 开发简单的树莓派单片机模块，让读者熟悉 Raspberry 和 Python。本书介绍 Linux 的最常用命令和 Python 的常用模块，并举实例详细讲解。

本书共 8 章，涵盖的主要内容有 Linux 和 Raspberry 简介、Raspberry 安装配置、Raspberry 开发利器、Raspberry 常用服务、Raspberry 常用功能、Raspberry GPIO、Raspberry 开门报警器实战、Raspberry 移动小车实战。本书所有源代码已上传网盘供读者下载使用。

本书内容丰富，实例典型，实用性强，适合树莓派初学者、物联网和智能家居开发人员，以及高等院校和培训学校相关专业的师生阅读。

图书在版编目（CIP）数据

树莓派开发从零开始学：超好玩的智能小硬件制作书 / 胡松涛编著. — 北京：清华大学出版社，2016
ISBN 978-7-302-43265-4

I. ①树… II. ①胡… III. ①软件工具－程序设计 IV. ①TP311.56

中国版本图书馆 CIP 数据核字(2016)第 044182 号

责任编辑： 夏非彼
责任校对： 闫秀华
责任印制：
出版发行： 清华大学出版社　　**地　址：** 北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编：** 100084
社 总 机： 010-62770175　　**邮　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 清华大学印刷厂
经　销： 全国新华书店
开　本： 190mm×260mm　　**印　张：** 13　　**字　数：** 333 千字
版　次： 2016 年 4 月第 1 版　　**印　次：** 2016 年 4 月第 1 次印刷
印　数： 1~3000
定　价： 69.00 元

产品编号：067082-01

前　言

随着计算机硬件的急剧微型化和物联网的快速发展，出现了越来越多的微型计算机，而树莓派（Raspberry Pi，本书简称为“Raspberry”）就是其中的佼佼者。Raspberry Pi 是一款针对电脑业余爱好者、教师、小学生以及小型企业等用户的迷你电脑，预装 Linux 系统，体积仅信用卡大小，搭载 ARM 架构处理器，运算性能和智能手机相仿。Raspberry 默认的操作系统还是 Linux，其他的微型计算机的操作系统大多也是嵌入式的 Linux。目前普通大众对 Linux 了解不多，即使有好的硬件设备也难以发挥它的功能。

网络上讲解 Raspberry 的帖子不少，要么语言不详，要么因为软硬件的升级而不再适用。本书是以实战为主旨，一步步地从安装系统开始，让读者熟悉 Linux、使用 Linux、喜欢 Linux，并安排了实战项目指导读者对 Raspberry 进行开发，开发中使用了 Python 脚本语言，会让读者眼界大开。

本书共 8 章，前面 5 章是 Raspberry 开发基础，第 6 章包括 LED 呼吸灯、蜂鸣器、超声波模块 3 个小实例，第 7 章为智能开门报警器实例，第 8 章为移动小车实例。没有任何 Linux 基础的读者，建议从第 1 章顺次阅读并演练每一个实例。有一定 Linux 基础的读者，可以根据实际情况有重点地选择阅读各个模块和项目案例。

本书特色

1．附带全部源代码，提高学习效率

为了便于读者理解本书内容，作者已将所有源代码上传到网络，供读者下载使用。读者通过源代码学习开发思路，优化代码。

2．涵盖 Linux 的安装配置和 python GPIO 的开发

本书涵盖 Linux 和 Raspberry 简介、Raspberry 安装配置、Raspberry 开发利器、Raspberry 常用服务、Raspberry 常用功能、Raspberry GPIO、Raspberry 开门报警器实战、Raspberry 移动小车实战。

3．对 GPIO Python 开发作了原理上的分析

本书在实战开发前对开发原理做出了详细的讲解，便于读者理解思路及代码的运行。

4．模块驱动，应用性强

本书提供了 3 个最简单的模块开发以方便读者自学，且这些模块可以组合应用成复杂的实际项目，具有超强的实用性。

5．项目案例典型，实战性强，有较高的应用价值

本书最后两章提供了 2 个项目实战案例，具有很高的应用价值和参考性，而且这些实例都是通过前面的基础讲解组合应用，便于读者融会贯通地理解本书所介绍的技术。这些案例稍加修改，便可用于实际项目开发中。

本书读者

- 树莓派开发初学者
- 单片机开发初学者
- 物联网开发人员
- 智能家居开发人员
- 高校和培训学校相关专业的师生

本书由胡松涛主笔，其他参与编写的有宋士伟、张倩、周敏、魏星、邹瑛、王铁民、殷龙、李春城、张兴瑜、马新原、李柯泉、林龙、赵殿华、牛晓云。

代码下载

本书源代码下载地址如下：

http://pan.baidu.com/s/1nuvxVi5

如果下载有问题，请电子邮件联系 booksaga@163.com，邮件主题为“树莓派”。

编　者

2016 年 1 月

目 录

第 1 章 Linux和Raspberry的简介

Raspberry Pi（中文名为“树莓派”，简写为 RPi，或者 RasPi/RPi）是为学生计算机编程教育而设计，只有信用卡大小的卡片式电脑，其系统基于 Linux。树莓派由注册于英国的慈善组织“Raspberry Pi 基金会”开发，Eben・Upton（埃・厄普顿）为项目带头人。Raspberry 外形只有信用卡大小，却具有电脑的所有基本功能。

本章主要内容包括：

- Linux 简介
- 了解 Raspberry 的知识
- 为 Raspberry 挑选合适的版本

1.1 Linux 前世今生

在了解 Raspberry 之前就不得不先了解一下 Linux。毕竟 Raspberry 默认的操作系统就是基于 Linux 的。本节简单地说明 Linux 的发展情况、目前流行的 Linux 版本及特点。

1.1.1 Linux 的诞生

Linux 是一套类 Unix 系统（Unix-like），是 Unix 的一种。它控制整个系统基本服务的核心程式 Kernel，是由美籍芬兰人 Linus Torvalds（2010 年入美国籍）于 1991 年带头开发出来的，Linux 这个名称便是以 Linus's Minix 来命名的。

Linus 选择用 GPL（General Public License）的方式来发行这份程序，这个版权允许任何人以任何形式散发、修改 Linux 的原始程序。换句话说，Linux 实际上是“免费的”。使用者在网络上就可以下载到 Linux 的原始程序，并随心所欲地散发与更改。在网络上日渐盛行以及 Linux 开放自由的版权之下，吸引了无数电脑高手投入开发、改善 Linux 的核心程序，使得 Linux 的功能日渐强大。今天我们可以在网络上免费下载 Linux 使用，这都是因为 Linux 是 GPL 版权的缘故。

Linux 实际上只是一份内核程序，它并不是操作系统。我们常说的 Linux，例如 Debian、CentOS、Fedora、Arch Linux、Gentoo、RHCE、Ubuntu、Deepin Linux、Rad Flag、StartOS 等等发行版本都是以 Linux kernel 为核心，加以必要的应用程序组合而来的。

1.1.2　Linux 的发行版本

Linux 的发行版本很多，无法统计具体数量。每天都有 Linux 发行版本诞生、消失。它们有很多都用于特殊场合，比如用于自启动光盘的 Kanotix Linux，用于教育方面的 EduLinux，用于 Network 检测的 Kali Linux（以前叫 Back Track）……

Linux 的发行版本可以大体分为两类：一类是商业公司维护的发行版本，一类是社区组织维护的发行版本。前者以著名的 RedHat 为代表，后者以 Debian 为代表。

1. Redhat

Redhat，应该称为 RedHat 系列，包括 RHEL（RedHat Enterprise Linux，也就是所谓的 RedHat Advance Server 收费）、Fedora（由原来的 RedHat 桌面版本发展而来，免费）、CentOS（RHEL 的社区克隆版本，免费）。RedHat 应该说是在国内使用人群最多的 Linux 版本，甚至有人将 RedHat 等同于 Linux。这个版本的特点就是使用人群数量大，资料非常多，言下之意就是如果你有什么不明白的地方，很容易找到人来问，而且网络上的一般 Linux 教程都是以 RedHat 为例来讲解的。RedHat 系列的包管理方式采用的是基于 RPM 包的 YUM 包管理方式，包分发方式是编译好的二进制文件。稳定性方面 RHEL 和 CentOS 非常好，适合于服务器使用，但是 Fedora 的稳定性一般，最好只用于桌面应用。

2. Debian

Debian，或者称 Debian 系列，包括 Debian 和 Ubuntu 等。Debian 是社区类 Linux 的典范，是迄今为止最遵循 GNU 规范的 Linux 系统。Debian 最早由 Ian Murdock 于 1993 年创建，分为三个版本分支（Branch）：Stable、Testing 和 Unstable。其中，Unstable 为最新的测试版本，包括最新的软件包，但是也有相对较多的 Bug，适合桌面用户。Testing 的版本都经过 Unstable 中的测试，相对较为稳定，也支持了不少新技术（比如 SMP 等）。而 Stable 一般只用于服务器，上面的软件包大部分都比较过时，但是稳定性和安全性都非常地高。Debian 最具特色的是 apt-get /dpkg 包管理方式，其实 RedHat 的 YUM 也是在模仿 Debian 的 APT 方式，但在二进制文件发行方式中，APT 应该是最好的了。Debian 的资料也很丰富，有很多支持的社区，有问题求教也有地方可去。

3. Ubuntu

Ubuntu，严格来说不能算一个独立的发行版本，Ubuntu 是基于 Debian 的 Unstable 版本加强而来，可以这么说，Ubuntu 就是一个拥有 Debian 所有的优点，以及自己所加强的优点的近乎完美的 Linux 桌面系统。根据选择的桌面系统不同，有三个版本可供选择：基于 Gnome 的 Ubuntu、基于 KDE 的 Kubuntu 以及基于 Xfce 的 Xubuntu。特点是界面非常友好，容易上手，对硬件的支持非常全面，是最适合做桌面系统的 Linux 发行版本。

4. Gentoo

Gentoo，伟大的 Gentoo 是 Linux 世界最年轻的发行版本，正因为年轻，所以能吸取在她之前的所有发行版本的优点，这也是 Gentoo 被称为最完美的 Linux 发行版本的原因之一。

5. FreeBSD

需要强调的是：FreeBSD 并不是一个 Linux 系统！但 FreeBSD 与 Linux 的用户群有相当一部分是重合的，二者支持的硬件环境也比较一致，所采用的软件也比较类似，所以可以将 FreeBSD 视为一个 Linux 版本来比较。

FreeBSD 拥有两个分支：Stable 和 Current。顾名思义，Stable 是稳定版，而 Current 则是添加了新技术的测试版。FreeBSD 采用 Ports 包管理系统，与 Gentoo 类似，基于源代码分发，必须在本地机器编译后才能运行，但是 Ports 系统没有 Portage 系统使用简便，使用起来稍微复杂一些。FreeBSD 的最大特点就是稳定和高效，是作为服务器操作系统的最佳选择，但对硬件的支持没有 Linux 完备，所以并不适合作为桌面系统。

1.1.3 Linux 的将来

Linux 的应用范围很广，可以说人类生活中处处都有 Linux。

1. 服务器

最常见的 Linux 应用是服务器。多年来，Linux 一直是超级计算机领域里的王者。在 Linux 企业级终端用户峰会（Linux Enterprise End-User Summit）上最新一期的世界最快超级计算机排行榜出炉，在世界最快超级计算机 500 强排行中，Linux 不仅占据主导地位，且将对手远远甩在身后。

同时它还有将其他对手挤出 500 强名单之势。在世界上 500 台最快的计算机里，强劲的开源操作系统 Linux 占了其中的 485 个位子，再创新高。换句话说，世界上最快的计算机里 97%是基于 Linux 的。

剩下的 15 台计算机里有 13 台运行 Unix 系统。这些计算机均运行 IBM Power 处理器，运行 IBM AIX 操作系统。其中最快的是英国的天气预测系统 ECMWF，在该榜单里排名第 60 位。

2. 嵌入式 Linux

嵌入式 Linux 是以 Linux 为基础的嵌入式作业系统，它被广泛应用在移动电话、个人数字助理（PDA）、媒体播放器、消费性电子产品以及航空航天等领域中。嵌入式 Linux 是将 Linux 操作系统进行裁剪修改，使之能在嵌入式计算机系统上运行。嵌入式 Linux 既继承了 Internet 上无限的开放源代码资源，又具有嵌入式操作系统的特性。我们使用的 Android 手机就是基于嵌入式 Linux。电视机顶盒基于嵌入式 Linux。路由器、交换机基于嵌入式 Linux。Play Station 基于嵌入式 Linux。车载导航系统基于嵌入式 Linux……可以说嵌入式 Linux 在我们周围无处不在。

3. Android

对，没错。目前最流行的手机操作系统 Android 同样也是基于 Linux 系统。虽然它只使用了 Linux 的内核，并对内核进行了一些必要的裁剪，但毫无疑问 Android 同样出生于 Linux，并依附 Linux 吸取营养茁壮成长。

4. Desktop

Linux 同样运行于桌面。不可否认 Linux 桌面使用率很低，但它的优秀同样是无须质疑的。Linux

桌面发行版很多，几乎每天都有数个版本诞生、消失。针对不同的用户，不同的使用环境都有相应的发行版本。有针对小存储设备的 Tiny Linux，有适合儿童使用的 OIMO Linux，有针对中国用户的麒麟 Linux，有专门用于网络检测的 Kali Linux，它的前身是大名鼎鼎的 BackTrack。

使用率最高的一般是 Debian、Ubuntu、Fedora、SUSE……它们虽然目前的市场占有率不高，但潜力强大不容小窥。


注 意

日常生活中最常用的桌面 OS 基本都是 Windows 或 IOS，Linux 桌面极为少见。对于 Linux，大部分人都是只在此山中、云深不知处的感觉。实际上 Linux 已经深入到我们生活之中不可分割。如果只是使用计算机可以无视 Linux，想要了解计算机就不得不了解 Linux。

1.2 深度剖析 Raspberry

Raspberry 自 2012 年发售以来，现在已经是第 2 代了。升级后的 Raspberry 性能增强了很多。按照这样的增长速度，有理由相信在不久的将来，Raspberry 性能能够追平一般的家用 PC。本章简单地介绍 Raspberry 的硬件配置及配件。

1.2.1 Raspberry Pi 的诞生

Raspberry 由注册于英国的慈善组织“Raspberry Pi 基金会”开发，Eben • Upton/埃 • 厄普顿为项目带头人。2012 年 3 月，英国剑桥大学埃本 • 阿普顿（Eben Epton）正式发售世界上最小的台式机，又称卡片式电脑，外形只有信用卡大小，却具有电脑的所有基本功能，这就是 Raspberry Pi 电脑板，中文译名树莓派。这一基金会以提升学校计算机科学及相关学科的教育，让计算机变得以有趣为宗旨。基金会期望这一款电脑无论是在发展中国家还是在发达国家，会有更多的其他应用不断被开发出来，并应用到更多领域。在 2006 年 Raspberry 早期概念是基于 Atmel 的 ATmega644 单片机，首批上市的 10000“台”Raspberry 的“板子”，由中国台湾和大陆厂家制造。

Raspberry1 是一款基于 ARM 的微型电脑主板，以 SD 卡为内存硬盘，卡片主板周围有两个 USB 接口和一个网口，可连接键盘、鼠标和网线，同时拥有视频模拟信号的电视输出接口和 HDMI 高清视频输出接口，以上部件全部整合在一张仅比信用卡稍大的主板上，具备所有 PC 的基本功能，只需接通电视机和键盘，就能执行如电子表格、文字处理、玩游戏、播放高清视频等诸多功能。Raspberry Pi 1 B 款只提供电脑板，无内存、电源、键盘、机箱或连线。

Raspberry 的生产是通过有生产许可的三家公司：Element 14/Premier Farnell、RS Components 及 Egoman 完成。这三家公司都在网上出售 Raspberry。

Raspberry1 配备一枚 700MHz 博通出产的 ARM 架构 BCM2835 处理器，256MB 内存（B 型已升级到 512MB 内存），使用 SD 卡当作储存媒体，且拥有一个 Ethernet 有线网卡接口，两个 USB 接口，以及 HDMI（支持声音输出）和 RCA 端子输出支援。Raspberry Pi 1 只有一张信用卡大小，体积大概是一个火柴盒大小，可以执行像雷神之锤 III 竞技场的游戏和进行 1080p 影片的播放。操作系统采用开源的 Linux 系统，比如 Debian、ArchLinux，自带的 Iceweasel、KOffice 等软件能够满足基本的网络浏览、文字处理以及计算机学习的需要，分 A、B 两种型号。

1.2.2 Raspberry 家族

目前最新的 Raspberry 是 Raspberry2。Raspberry 各个型号的参数如表 1-1。

表 1-1 Raspberry 参数

项目	A+型	B 型	B+型	2 代 B 型
SoC	Broadcom BCM2835			Broadcom BCM2836
CPU	ARM1176JZF-S 700MHz			ARM Cortex 900MHZ 4 核
GPU	Broadcom VideoCore IV，OpenGL ES2.0，1080p 30 h.264/MPEG-4 AVC			
内存	256MB	512MB	1GB	
USB2.0	1	2	4	
视频	RCA 视频输入（仅 1 代 B 型），支持 PAL，NTSC，HDMI			
音频	标准 3.5mm 插孔，HDMI			
SD 卡接口	TF 卡，1 代 B 型的使用 SD 卡，也可用 TF 卡装入 SD 卡适配器中使用			
网络接口	没有（通过 USB）	10/100 以太网网卡（RJ45 接口）		
扩展接口	40	26	40	
额定功率		700mA/3.5W	600mA/3.0W	1000mA/5W
电源输入	5V，通过 MicroUSB 或者 GPIO 引脚			
总体尺寸	65mm×56mm	85.60mm×53.98mm	58mm×56mm	
OS	Linux，2 代 B 型还支持 Windows10			

短短几年，Raspberry 的配置翻了一倍。虽然性能目前无法跟 PC 相比，但只是做私人服务器、桌面、代码开发、HTPC……家庭应用是足够了。


注 意

Raspberry 就相当于（不是等同于）集成了 CPU、显卡、内存的微型 PC 主板，只不过这块主板是 Arm 构架的。

1.3 Raspberry 配件选择

独木难成林，光秃秃的一块板子再怎么逆天也是没用的，没有其他配件的配合照样没用。下面我们来熟悉一下 Raspberry 的外设设备。

1.3.1 Raspberry 必要设备

虽说最简单的配置只需要一个 Raspberry，一个 TF 卡，一根充电线，一个充电头即可。可这样的“低配版”的实用效果如何那是可想而知了。下面就来说说 Raspberry 的必要设备及挑选要求。

1. TF 卡

TF 卡官方要求的只是 4G 以上，Class4 以上就可以了。品牌未做要求。但现在网上一般都推荐闪迪。我用的其他品牌也没事。为安全起见还是用大家推荐的品牌好了。容量不必太大，

8GB~16GB 就可以了，4G 的 TF 卡装了系统就不剩什么了。至于速度 Class4 是最低要求，Class10 当然更好。

2. 充电线

Raspberry 所需的充电线是 Micro USB 通用充电线，就是一般 android 手机的电源线。可以找一根备用的手机充电线来用，但建议还是买一根带开关的充电线。Raspberry 没有开关机按钮，只有通过连接/断开电源来开关机。每次都拔充电头比较麻烦。

3. 充电头

充电头也可以用手机充电头，但要求是 5V/2A。一般手机充电头都是 1.5A 的，如果电流不足可能会出现各种问题。所以，如果没有合适的充电头，还是买个符合标准的吧。

4. 散热片（风扇）

散热片是必不可少的设备了。如果不想玩 Raspberry 正兴奋的时候黑屏，还是给它配上两个吧。只要不是 7×24 开机，散热片就足矣。也可以用小风扇，可风扇需要接电源，比较麻烦。如果没有特殊要求还是用散热片吧。

5. 外壳

虽说把 Raspberry 放到桌上或是用个木夹把它夹住也不是不行。但为了美观和安全着想，最好给它配个外壳。配个一般的亚克力外壳就行，避免其他小物件碰到 Raspberry 而短路。

6. HDMI 线（HDMI 转 VGA 线）

HDMI 线可以说是 PC 的标准配置了，要是不嫌麻烦，就用 PC 上的那根吧。如果显示器没有 HDMI 接口，那就只有配一根 HDMI 转 VGA 线了。VGA 接口几乎每个显示器都有。

1.3.2 Raspberry 非必要设备

以下这些设备看各自的研究方向和要求选用。

1. PC 配件

免驱 USB 无线网卡、USB 集线器、非 PS2 接口的键盘鼠标、小型显示器、红外、蓝牙适配器……这些有当然更好，没有也没关系。无非就是方便顺手的问题。

2. 存储器

不管是大容量的硬盘配硬盘盒还是大容量的移动硬盘，来一个吧。光靠 TF 卡的容量也只能装个系统。如果想让 Raspberry 发挥更大的作用，还是得加上个大的存储器。当然，如果没有也行。

3. 各种传感器

传感器是扩展 Raspberry 时需要的。检测烟雾，就得有烟雾气敏传感器。测距避障，就得有超声波传感器。检测光线，就得有光敏传感器。 检测温度湿度，就得有温度传感器和湿度传感器……

4. 面包板，杜邦线

如果不想研究硬件，这个是可以略过的。或者可以找别的物品替代。

5. 其他设备

二极管，三极管，电阻……不想研究硬件的可以略过。

1.4 Raspberry OS 的选择

适合 Raspberry 的发行版本很多，足以应付不同人群的挑选。本节列出最常见的 Raspberry 的操作系统，简单说明各个发行版本的适用范围及特点，以便于大家挑选最适合自己的系统。

1.4.1 Raspberry 官网推荐 OS

适合 Raspberry 的发行版本有很多，很难一一列数，在这里只列出 Raspberry 官方推荐的几个版本。当然，非官方的 Raspberry 版本同样优秀。具体需要哪个版本，还要看各自的用途。笔者选择的是 RaspBian，也是使用最广泛的 Raspberry 操作系统。

1. NOOBS

官方推荐的系统，可以多系统引导（包含 RaspBian、Arch、OpenELEC、RaspBMC……），是个非常好用的多系统引导器。它本身就含有操作系统的全部文件，可以完全不依赖网络直接安装系统。只要记得安装完成后更新系统就行。

2. NOOBS LITE

官方推荐的系统，可以多系统引导。它不含操作系统的文件，纯粹是个引导器，需要依赖网络。如果网络条件非常好的情况下，选它也不错。

3. RASPBIAN

RaspBian 是专门用于 ARM 卡片式计算机 Raspberry Pi 的操作系统。RaspBian 系统是 Debian 7.0/wheezy 的定制版本，得益于 Debian 从 7.0/wheezy 开始引入的带硬件浮点加速的 ARM 架构（armhf），Debian 7.0 在树莓派上的运行性能有了很大提升，RaspBian 默认使用 LXDE 桌面，内置 C 和 Python 编译器。

RaspBian 是 Debian 为 Raspberry 定制的版本。基本上和 Debian 是一模一样的。Debian 使用的人数很多，稳定性好，符合 POSIX（Portable Operating System Interface）标准，文件系统规范，安全稳定。如果只是日常需要，几乎不需要更新。国内的更新源多，要知道 Linux 非常依赖网络，软件安装、系统更新都需要网络支持。以国内的网络条件来说，还是选择一个国内源比较多的发行版本比较方便。

此外，其他的第三方版本各有侧重的方面，RaspBian 可以说是使用最平衡的版本。等 RaspBian 使用熟练了，需要其他方面支持的时候再换其他的版本。



注 意

Debian 可以说是使用跨度最大的 Linux 版本。它的软件丰富，系统稳定，几乎支持所有的硬件架构。不管是初学者还是资深用户使用 Debian 都非常顺手，它是最好的社区版 Linux。这也许就是 Raspberry 官方为什么首推 RaspBian 的原因吧。个人认为国内流行 RH 系列主要是跟其商业推广有关，如果个人用户还是 Debian 系列比较合适。

1.4.2 官方推荐的第三方 OS

第三方的 OS 都有强烈的自身风格。它们往往对某一方面的支持非常好，但对其他方面就稍微差一点点。如果需要的只是某一方面的功能，那么选择第三方的 OS 是非常方便的。尤其是由官方推荐的第三方 OS，在其特定的功能方面是无须质疑的。

1. UBUNTU MATE

使用的是 Ubuntu 的 ARM 版本，Gnome2 桌面。使用过 Ubuntu 的用户会非常熟悉这个版本。这个版本非常适合做桌面，而且 Ubuntu 的社区支持非常丰富。不愁有问题没地方问。

2. OSMC

OSMC 是 Open Source Media Cente 的缩写。一个开源的媒体中心。RaspBian 配合 XBMC，深度定制，不包含其他意义不大的包，最大程度发挥树莓派的视频处理能力。它默认是英文的版本，有国内开发的 Raspbmc，这个更适合国情。如果只是想做 HTPC，这个版本是非常合适的。

3. Snappy Ubuntu Core

Ubuntu 的一个版本。Snappy Ubuntu Core 是面向智能设备的最新平台，可以运行存储在本地或依赖于云端的相同软件，而后者的最大好处就是可以让使用者避免频繁地定期升级。

4. OpenELEC

OpenELEC 是 Open Embedded Linux Entertainment Center 的缩写，从字面上理解为开源嵌入式 Linux 娱乐中心，功能没有 Raspbmc 那么强大，但是对于普通的高清播放来说，这个完全足够用了。

5. RISC OS

RISC OS 不是 Linux，但它是一个实时系统。可惜很多软件都只能编译使用，有些不方便，对于有嵌入式开发经验的人士，会有一些注重实时性的应用。非专业人士不推荐使用.

1.4.3 其他的 OS

除了官方 OS 和官方推荐的 OS 外，其他的 OS 发行方也为 Raspberry 准备了配套的 OS。它们也许没有那么强烈的自身特点，但对那些只使用某个 OS 的死忠用户而言，它们将是最顺手的选择。

1. ArchLinuxARM

ArchLinuxARM 是 ArchLinux 的 ARM 版本。以轻量出名，使用 pacman 可以快速找到自己的

软件。ArchLinux 也许不是最好的系统，但它一定是最方便的系统。

2. Pidora

Pidora 是 Fedora 的 ARM 版本。该系统是其专门为树莓派迷你计算机开发的基于 Fedora remix 的系统，Pidora 18 完全基于 ARMv6 架构的 Fedora 软件包，几乎所有 Fedora 软件包都可以在 Pidora 上通过 yum 安装。Pidora 的包是从 Fedora 官方资源库直接构建，同时 Pidora 包含树莓派特定的一些配置模块，包括默认 SD 卡映像，库和外部硬件设备（GPIO/I2C/SPI 接口等）的支持都已包含。Fedora 的赫赫威名无须任何宣传，足以在此占领一方。

3. Windows 10

这个系统只有在 Raspberry2 才能使用。Windows 10 系统的优秀就不用再多介绍了。将 Windows 系统融入 Raspberry 可以说是一次里程碑式的进步。虽然说目前它还很难与 PC 上的 Windows 10 相媲美。但技术总是不断进步的，不是吗？非常期待 Mac OS 也能加入 Raspberry 的大家庭。

第 2 章
◂Raspberry的安装配置▸

对 Raspberry 的硬件和操作系统有了一定的了解，下面应该安装系统了。本章的目的是完全从零开始，一步步地下载所需软件，正确地安装配置 Raspberry 并将其备份还原。让它能发挥自己的作用。

本章主要内容包括：

- 下载 Raspberry 的系统
- 配置 RaspBian
- 实现远程无密码登录
- 系统的备份和还原

2.1 从零开始安装配置 Raspberry

本节将下载安装系统所需的软件和 Raspberry 的操作系统，演示在不同系统中写入操作系统到 SD 卡，并做好配置系统前的准备工作。

2.1.1 下载 Raspberry 的系统

鉴于 RaspBian 的平衡和广泛，这里选择使用 RaspBian 系统。现在就来下载 RaspBian 系统。

（1）首先我们百度一下“RaspBian 官网”，得到 RaspBian 的官网网址 www.raspbian.org。

（2）打开这个网站，单击左侧的 Images 超链接，在 raspberry pi foundation Raspbian Images 一项中得到了 RaspBian 的下载页面 www.raspberrypi.org/downloads/。

（3）选择 Raspbian debian weezy 下载，大概有 1GB 左右，请耐心下载。

这里，我选择的是 http 下载，得到 Raspbian_latest 文件，然后解压缩，得到 2015-05-05-raspbian-wheezy.img 文件。在这里下载的文件应该是安全的，如果还不放心，下载地址下方有校验码，可以自行校验一下。好了，现在可以开始写入系统了。

2.1.2 Windows 下安装 RaspBian

因为绝大部分人都使用 Windows 系统，所以我们先从 Windows 系统开始。

Windows 系统中使用的写入软件是 Win32DiskImager。它的作用是将一个文件写入 U 盘。如果

有其他功能相似的软件都可以使用。当然，既然这个软件可以用来安装 RaspBian，同样它也可以用来安装 Linux 和 Windows。具体方法，请咨询百度，下面开始具体操作。

（1）先将 TF 卡插入读卡器中，将读卡器插入电脑的 USB 接口。打开 Win32 Disk Imager 窗口，如图 2-1 所示。

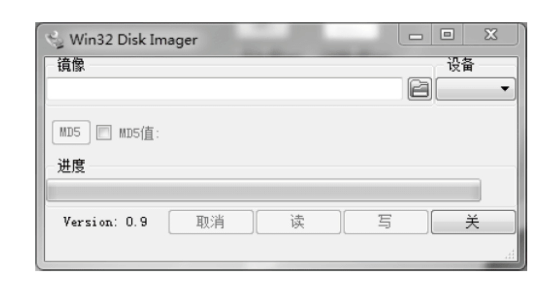


图 2-1　Win32 disk Imager 窗口

（2）在“设备”下面的下拉框选择 U 盘的盘符。单击文件夹图标，选择刚才下载并解压后得到的 img 文件。然后单击“写”按钮，进行写入。

如果写入失败，没关系。按照刚才的步骤，重新再来一次。如果多次写入不成功，请检查一下 TF 卡是否损坏，更换 TF 卡后再次写入。

（3）写入完成后，拔出读卡器，取出 TF 卡，插入 Raspberry。

2.1.3　Linux 下安装 RaspBian

前面铺垫了那么多 Linux 的知识，实际上就是建议大家在 Linux 下使用 Raspberry。这里介绍一下 Linux 下的安装步骤。

（1）先将 TF 卡插入读卡器，再将读卡器插入电脑的 USB 接口。

进入系统桌面后，打开 Terminal（Linux 的版本太多，桌面环境也不一样，具体怎样打开 Terminal，请自行百度一下）。或者直接按 Ctrl + Alt + F2 组合键，如图 2-2 进入控制台。

```
Debian GUN/Linux 7 debian7 tty2

debian7 login:
```

图 2-2　Linux 控制台

Linux 默认情况下有 7 个控制台，快捷键也就是 Ctrl + Alt + F1~F7。一般情况下按 Ctrl + Alt + F7 组合键进入图形界面。但也有把图形界面放在 Ctrl + Alt + F1 的，其他的都是 Consel 字符界面。所以，Ctrl + Alt + F2 最安全。

输入用户名、密码登录。如果能用 root 登录，尽量使用 root。如果没有 root 权限，那至少登录用户在 sudoers 文件中，并且有相应的执行权限。执行命令时，在命令前加上 sudo。要知道一般用户是无法使用 dd 命令来操作磁盘的。在这里用 root 登录，如图 2-3 所示。

```
Debian GUN/Linux debian7 tty2

debian7 login: root
Password:
Last Login: Sun Jun 28 18:17:02 CST 2015 on tty2
Linux debian7 3.2.0-4-amd64 #1 SMP Debian 3.2.65-1 X86_64

The programs included with the Debian GNU/Linux system are free soft:ware;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyringt.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@debian7:~# _
```

图 2-3　root 登录

（2）使用 ls –l /dev/sd*命令查看 TF 卡磁盘位置，如图 2-4 所示。

```
ls -l /dev/sd*
```

```
root@debian7:~# ls -l /dev/sd*
brw-rw---T 1 root disk        8,    0 Jun 28 18:07 /dev/sda
brw-rw---T 1 root disk        8,    1 Jun 28 18:07 /dev/sda1
brw-rw---T 1 root disk        8,    2 Jun 28 18:07 /dev/sda2
brw-rw---T 1 root disk        8,    5 Jun 28 18:07 /dev/sda5
brw-rw---T 1 root floppy      8, 16 Jun 28 18:28 /dev/sdb
brw-rw---T 1 root floppy      8, 17 Jun 28 18:28 /dev/sd1
root@debian7:~# umount /dev/sdb1
umount: /dev/sdb1: not mounted
root@debian7:~# _
```

图 2-4　查看 TF 卡

一般的 SATA 硬盘都是以/dev/sd 开头的。如果用的是 IDE 硬盘，则是以/dev/hd*开头。

如果主机只有一块 SATA 硬盘，那么这块硬盘的标识就是/dev/sda。在此例中，主机只有一块 SATA 硬盘，所以读卡器中的 TF 卡被识别为/dev/sdb。

sda1 是 sata 硬盘的第一主分区，sda2 是 sata 硬盘的第二主分区。sda5 是 sata 硬盘的第一逻辑分区。同理，sdb1 是读卡器中 TF 卡的第一主分区。

执行命令：

```
umount /dev/sdb1  ~
```

这个命令的作用是卸载读卡器中 TF 卡的第一主分区。

因为有的 Linux 发行版本默认自动挂载 U 盘、读卡器等即插即用设备，所以执行 umount 命令以防万一。挂载了，就把读卡器分区卸载；没挂载，卸载一下也没什么影响，以防万一。

在这里，只有/dev/sdb1，所以就只执行了 umount /dev/sdb1。如果有 sdb2，sdb3……那就得继续执行 umount /dev/sdb2 umount /dev/sdb3……

（3）卸载了读卡器的分区，现在开始写入 RaspBian 系统到 TF 卡。先进入下载文件的分区。执行命令：

```
cd ~
```

进入如图 2-5 所示的下载文件所在目录。如果解压出来的 2015-05-05-raspbian-wheezy.img 在其他的目录，请进入该目录。

```
root@debian7:~# cd
root@debian7:~# ls
2015-05-05-raspbian-wheezy.img
root@debian7:~# 
```

图 2-5　进入工作目录

然后使用 dd 命令将 2015-05-05-raspbian-wheezy.img 写入磁盘中去，如图 2-6 所示，执行命令：

```
dd bs=4M if=2015-05-05-raspbian-wheezy.img of=/dev/sdb
```

```
root@debian7:~# cd
root@debian7:~# ls
2015-05-05-raspbian-wheezy.img
root@debian7:~# dd bs=4M if=./2015-05-05-raspbian-wheezy.img of=/dev/sdb
```

图 2-6　系统写入 TF 卡


注　意

这里 of 后面的参数是/dev/sdb，而不是/dev/sdb1，目的是将 img 文件写入整个磁盘，而不是磁盘的某个分区。

【dd 命令简介】

现在在 Linux 下，那么我们用 man dd 来查看一下 dd 的功能，如图 2-7 所示，执行命令：

```
man dd
```

按照 man 的解释：

```
DD(1)                                                          DD(1)

NAME
       dd - 转换和拷贝文件

摘要
       dd  [--help]  [--version]  [if=file]  [of=file] [ibs=bytes] [obs=bytes]
       [bs=bytes]  [cbs=bytes]  [skip=blocks]   [seek=blocks]   [count=blocks]
       [conv={ascii, ebcdic, ibm, block, unblock, lcase, ucase, swab, noerror,
       notrunc, sync}]
```

图 2-7　man dd

dd 命令的功能与 Win32DiskImager 比较相似，不过功能比 Win32DiskImager 更强大。dd 命令可以把文件写入磁盘、分区、文件，也可以把磁盘、分区、文件写入文件。

下面来看下 dd 命令的常用参数，如图 2-8 所示：

```
pi@raspberrypi ~ $ dd --help
Usage: dd [OPERAND]...
  or:  dd OPTION
Copy a file, converting and formatting according to the operands.

  bs=BYTES        read and write up to BYTES bytes at a time
  cbs=BYTES       convert BYTES bytes at a time
  conv=CONVS      convert the file as per the comma separated symbol list
  count=BLOCKS    copy only BLOCKS input blocks
  ibs=BYTES       read up to BYTES bytes at a time (default: 512)
  if=FILE         read from FILE instead of stdin
  iflag=FLAGS     read as per the comma separated symbol list
  obs=BYTES       write BYTES bytes at a time (default: 512)
  of=FILE         write to FILE instead of stdout
  oflag=FLAGS     write as per the comma separated symbol list
  seek=BLOCKS     skip BLOCKS obs-sized blocks at start of output
  skip=BLOCKS     skip BLOCKS ibs-sized blocks at start of input
  status=noxfer   suppress transfer statistics
```

图 2-8　dd –help

最常用的选项如下：

- if =输入文件（或设备名称）
- of =输出文件（或设备名称）
- ibs = bytes，一次读取 bytes 字节，即读入缓冲区的字节数
- obs = bytes，一次写入 bytes 字节，即写入缓冲区的字节数
- bs = bytes，同时设置读/写缓冲区的字节数（等于设置 ibs 和 obs）


注　意

详细的解释请参考 man dd。

（4）dd 命令执行完毕后，拔出读卡器，取出 TF 卡，插入到 Raspberry。

2.1.4　Mac OS 下安装 RaspBian

国内目前苹果电脑也变得越来越流行，所以我们也说一下 Mac 下的安装。

（1）首先还是打开命令行。

单击“应用程序菜单”，单击“实用工具”子菜单，单击“终端”选项。

（2）Mac OS 是类 Unix 系统，而 Linux 脱胎继承于 Unix。所以，有很多命令都是通用的。不同的是在 Linux 下磁盘位置是/dev/sd*或者是/dev/hd*，U 盘的位置是/dev/sd*。在 Mac OS 下磁盘位置是/dev/disk*。/dev/rdisk1 是原始字符设备，也就是 U 盘。所以只需要将 Linux 的 dd 命令稍微地改装一下就可以了：

```
dd bs=4M if=2015-05-05-raspbian-wheezy.img of=/dev/rdisk1
```

好了将 RaspBian 系统写入磁盘后，下面可以开始配置系统了。


注　意

不管是用哪种方法安装 RaspBian，实质上都是将文件写入硬盘。所以如果有其他的硬盘写入软件可用，放心大胆地试吧。

2.2 RaspBian 基本配置

在安装 Windows 时，需要用户选择安装磁盘，选择时区……（当然这里说的是正常安装 Windows 的方法，Ghost 安装是不需要的。RaspBian 做好备份后，再次安装等同于 Ghost 安装，Windows 也可以不需要配置）安装 Linux 同样也要选择这些。本章目的在于了解 RaspBian 的安装步骤，熟悉 Raspi-config 配置选项，正确配置 Raspberry。

2.2.1 raspi-config 配置

将写入完毕的 TF 卡插入 Raspberry 中。在 USB 端口上连接好鼠标键盘（只能选用 USB 端口的鼠标键盘，我选用的是无线鼠标键盘，好处是占用的端口少），用 hdmi 线连接到显示器上，插上电源，Raspberry 就开始运行了。

第一次运行 Raspberry 时，会自动运行一个程序 raspi-config，如图 2-9 所示。这个程序用来设置 Raspberry 的基本选项。下面我们就来一一配置。

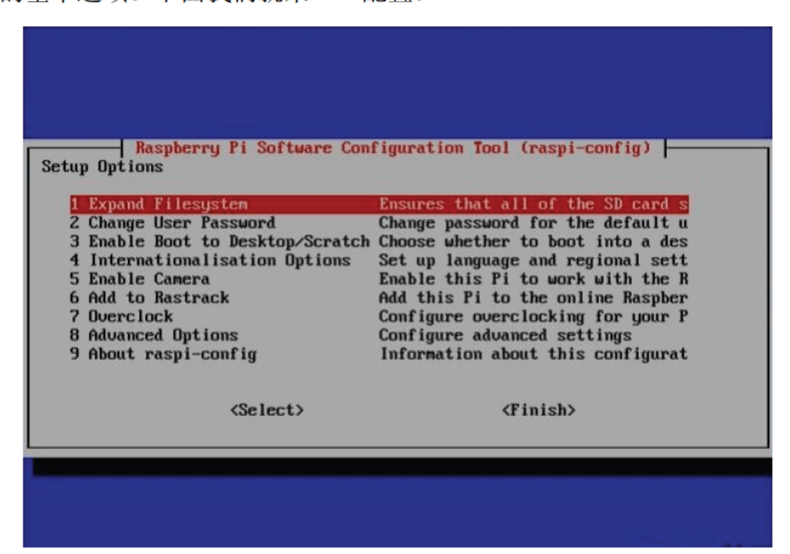


图 2-9 raspi-config 主界面

1. Expand Filesystem：Ensures that all of the SD card storage is available to the OS

扩展文件系统，如图 2-10。默认镜像写入 TF 卡后，根分区不会使用剩余的 TF 卡空间，也就是说不管你的 TF 卡有多大，系统只使用了 1GB 左右。剩下的空间都浪费，运行此选项后会把根分区扩展到整个 TF 卡，最大效率使用 TF 卡。如需要扩展，按 Enter 键确定。如果不需要，按 Esc 键返回 raspi-config 主界面。

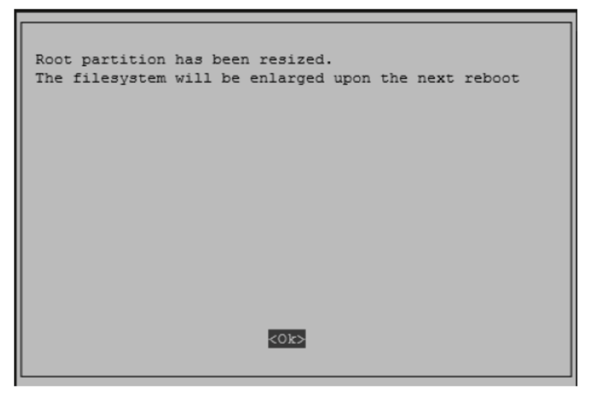


图 2-10　Expand Filesystem

2. Change User Password：Change password for the default user　（pi）

改变默认用户 pi 的密码，如图 2-11 所示。按 Enter 键后输入 pi 用户的新密码。Raspberry 默认的用户名 pi 的默认密码是 raspberry。



图 2-11　Change User Password

按 Enter 键，如图 2-12 所示。

```
Enter new UNIX password:
Retype new UNIX password:
```

图 2-12　连续 2 次输入密码

这里请 2 次输入新密码。按 Enter 键，如图 2-13 所示。

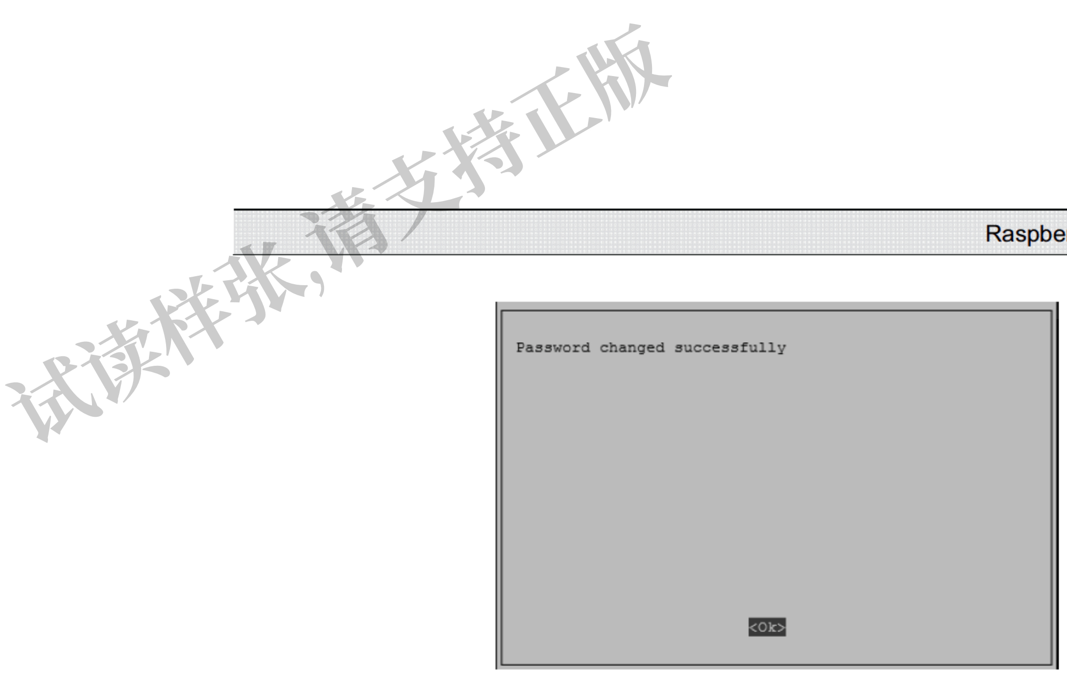


图 2-13　修改密码成功

显示修改密码成功后，按 Enter 键，回到 raspi-config 主界面。

3. Enable Boot to Desktop/Scratch：Choose whether to boot into a desktop evironmetn, scratch, or the command-line

启动时进入的环境选择，如图 2-14 所示。

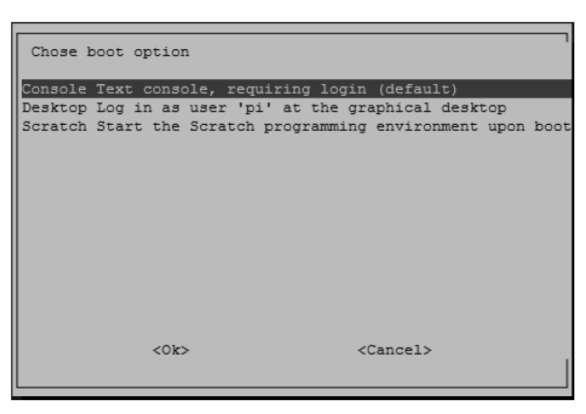


图 2-14　启动环境选择

- Console Text console, requiring login（default）：启动时进入字符控制台，需要进行登录（默认项）。
- Desktop Log in as user 'pi' at the graphical desktop: 启动时以用户 pi 登录 LXDE 桌面环境。
- Scratch Start the Scratch programming environment upon boot：启动时进入 Scratch 编程环境。

一般来说，选择 Desktop Log in as user 'pi' at the graphical desktop 比较方便。

以 Up 键（上箭头）Down 键（下箭头）移动光标，选择好后用 Tab 键跳至<OK>选项。按 Enter 键，回到 raspi-config 主界面。

4. Internationalisation Options：Set up language and regional settings to match your location

国际化选项，可以更改语言、时区、键盘布局，如图 2-15 所示。

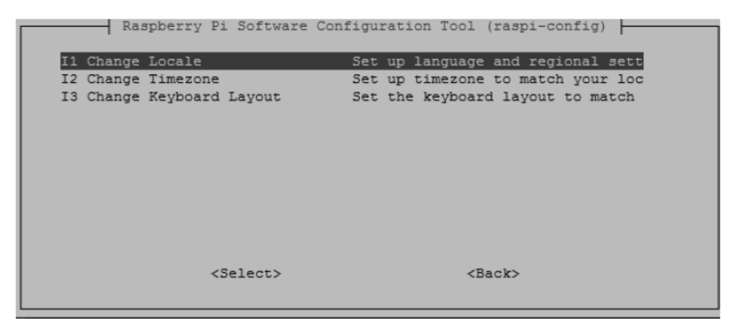


图 2-15　国际化配置主界面

- I1 Change Locale: 语言和区域设置，按 Enter 键进入选择，如图 2-16 所示。以 Up 键（上箭头）Down 键（下箭头）移动光标，用空格键选择。选择系统支持的字符编码 en_US.UTF-8、zh_CN.UTF-8。有这两个基本上就足够了，如有特殊需要，酌情选择。

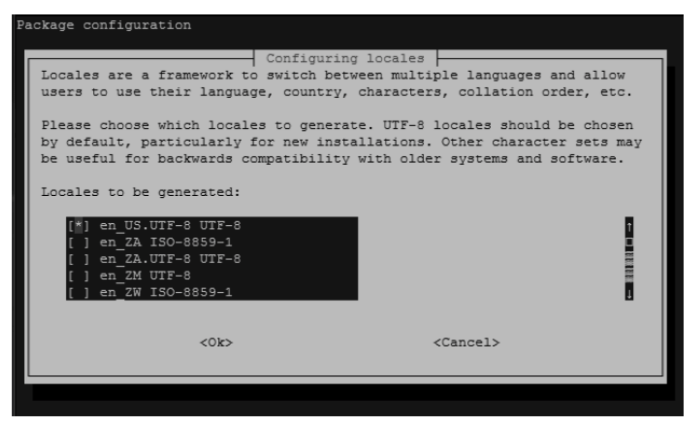


图 2-16　系统语言选择

- I2 Change Timezone：设置时区，如果不进行设置，PI 的时间就显示不正常。选择 Asia（亚洲），如图 2-17 所示。

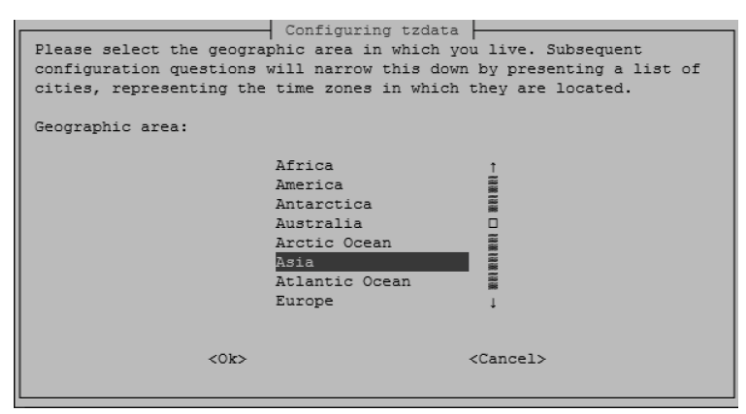


图 2-17　时区选择

- I3 Change Keyboard Layout：选择键盘布局，默认的键盘布局是英式的。需要修改一下。但在这里配置修改比较麻烦。可以执行命令：

```
nano -w /etc/default/keyboard
```

将其内容修改成下面的代码：

```
XKBMODEL="pc104"
XKBLAYOUT="us"
XKBVARIANT=""
XKBOPTIONS="terminate:ctrl_alt_bksp"

BACKSPACE="guess"
```

按 Ctrl + x 组合键，保存退出。如果文件被改动，会提示是否保存修改，输入“Y”，按 Enter 键保存。

5. Enable Camera：Enable this Pi to work with the Raspberry Pi Camera

启动 Pi 的摄像头模块，如果想启用，选择 Enable，禁用选择 Disable 就行了。

6. Add to Rastrack：Add this Pi to the online Raspberry Pi Map （Rastrack）

把你的 Pi 的地理位置添加到一个全世界开启此选项的地图。

7. Overclock：Configure overclocking for your Pi

超频，强烈建议不要更改，更改后会失去保修。

- None 不超频，运行在 700Mhz，核心频率 250Mhz，内存频率 400Mhz，不增加电压。
- Modest 适度超频，运行在 800Mhz，核心频率 250Mhz，内存频率 400Mhz，不增加电压。
- Medium 中度超频，运行在 900Mhz，核心频率 250Mhz，内存频率 450Mhz，增加电压 2V。
- High 高度超频，运行在 950Mhz，核心频率 250Mhz，内存频率 450Mhz，增加电压 6V。
- Turbo 终极超频，运行在 1000Mhz，核心频率 500Mhz，内存频率 600Mhz，增加电压 6V。

8. Advanced Options：Configure advanced settings

高级设置

- A1 Overscan：是否让屏幕内容全屏显示。
- A2 Hostname：在网上邻居或者路由器能看到的主机名称。
- A3 Memory Split：内存分配，选择给 GPU 多少内存。
- A4 SSH：是否运行 SSH 登录，强烈建议开启此选项，方便以后操作 Pi，有网络就行，不用开启屏幕了。
- A5 Device Tree：是否启用 Device Tree，略过。
- A6 SPI：是否默认启动 SPI 内核驱动，略过。
- A7 I2C：是否载入 I2C 总线模块，略过。

- A8 Serial：是否串行连接内核、shell，略过。
- A9 Audio：选择声音默认输出到模拟口还是 HDMI 口。
- A0 Auto：自动选择。

```
1 Force 3.5mm （'headphone'） jack
```

强制输出到 3.5mm 模拟口

```
2 Force HDMI
```

强制输出到 HDMI。

```
A7 Update
```

把 raspi-config 这个工具自动升级到最新版本。

9. About raspi-config：Information about this configuration tool

关于 raspi-config 的信息。
所有设置完毕后按 TAB 键，单击 Finsh 按钮后按 Enter 键保存。

2.2.2 网络配置

此时，Raspberry 基本上可以运行了。不过革命尚未成功，同志仍需努力。要想方便地使用 Raspberry，还有一些地方需要配置。首先我们需要配置的是网络，以有线网络为例。

将网线正确地插入到 rj45 端口，网络参数的配置文件是/etc/network/interfaces。



注 意

Linux 下最有名的文字编辑器是 nano 和 vi。几乎所有的 Linux 至少默认安装了其中的一种。个人推荐使用 vi 的扩展版本 vim。RaspBian 只安装 vi，没有默认安装 vim，不过没关系，暂时先用 nano 吧。它也挺不错的，跟 Windows 下的 notepad（记事本）很类似。
Linux 几乎所有的系统配置文件都在/etc 下。etc 源自于拉丁语中 etcetera，有零散的意思。

好了，现在我们来设置有线的网络连接。

（1）执行命令：

```
sudo cp /etc/network/interfaces /etc/network/interfaces.bak
sudo nano -w  /etc/network/interfaces
```

sudo 是 super user do 的缩写。它在此处的作用是以超级用户的权限来执行命令。第一条命令的作用是，在 interfaces 的目录下创建一个备份文件，以免文件破坏后无法恢复。第二条命令的作用是，用 nano 编辑器打开 interfaces 文件，如图 2-18 所示。

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
iface eth0 inet manual

auto wlan0
allow-hotplug wlan0
iface wlan0 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

auto wlan1
allow-hotplug wlan1
iface wlan1 inet manual
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

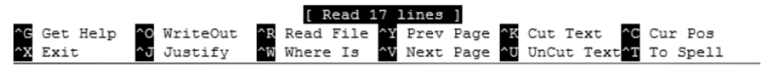


图 2-18 nano 编辑 interfaces

（2）自动运行 lo ，lo 即 localhost，就是 127.0.0.1。

```
auto lo
```

（3）回环地址

```
Iface lo  inet loopback
```

（4）设置网络接口 eht0 的 IP 获取方式 dhcp。eth0 是有线网络的第一个网络接口，第二个就是 eth1……wlan0 是无线网络的第一个网络接口，第二个就是 wlan1。

```
Iface eth0 inet dhcp
```

（5）再执行命令：

```
allow-hotplug eth0
```

意思是 eth0 网络接口允许热插拔，这里主要是配置有线网络，也就是 eth0，那么只需要留下系统的回环地址和 eth0 的配置就可以了。

最终接结果如下：

```
####    lo 是回环配置
auto lo
iface lo inet loopback

####    eth0 是第一个有线网卡
auto eth0
allow-hotplug eth0
#iface eth0 inet manual
iface eth0 inet static
address 192.168.2.11
netmask 255.255.255.0
gateway 192.168.2.1
```

```
####   wlan0 是第一个无线网卡
#auto wlan0
#allow-hotplug wlan0
#iface wlan0 inet manual
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

####   wlan1 是第二个无线网卡
#auto wlan1
#allow-hotplug wlan1
#iface wlan1 inet manual
```


注 意

单行注释只用在行首添加“#”符号。把所有不需要的行前面都添加一个“#”。如果实在是不需要，完全可以把这些注释行删除。

```
iface eth0 inet static
```

这种是设置 eth0 获取 IP 的方式。static 是指设置静态 IP。另外一种就是 dhcp（Dynamic Host Configuration Protocol，动态主机配置协议）是由系统分配 IP。如果需要设置成 dhcp，应该如下设置：

```
iface eth0 inet dhcp
```

设置成静态分配 IP 后，就必须给出网络地址、子网掩码和网关。DHCP 就不用了。

```
address 192.168.2.91
netmask 255.255.255.0
gateway 192.168.2.1
```

最后看最下面的 2 行，是 nano 的提示信息，如图 2-19 所示。

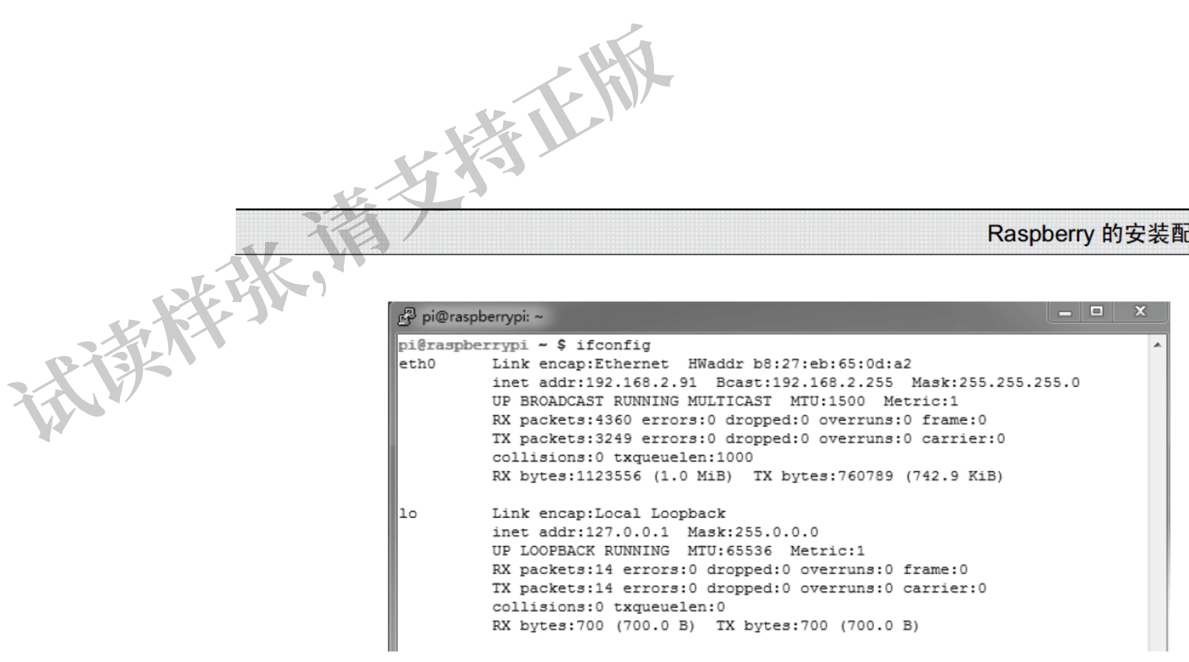
```
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

图 2-19　nano 提示信息

按 Ctrl + x 组合键，保存退出。如果文件被改动，会提示是否保存修改，输入 Y，按 Enter 键保存。好了，如果只需要有线网络，这样设置就足够了。然后重启网络服务，这个设置就生效了。

查看修改配置后的结果，如图 2-20 所示。

```
sudo /etc/init.d/networking restart
ifconfig
```



```
pi@raspberrypi ~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:65:0d:a2
          inet addr:192.168.2.91  Bcast:192.168.2.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4360 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3249 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1123556 (1.0 MiB)  TX bytes:760789 (742.9 KiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:14 errors:0 dropped:0 overruns:0 frame:0
          TX packets:14 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:700 (700.0 B)  TX bytes:700 (700.0 B)
```

图 2-20　ifconfig

网络配置成功，可以用 ping 命令自行检测一下。

2.2.3　无线网络配置

如果可以，尽量使用有线网卡。不是 Raspberry 的无线不给力，而是免驱支持 Raspberry 的无线网卡实在是不多。为了避免安装驱动的麻烦，还是使用有线网卡方便。

如果无线网卡是免驱的或者已经安装好驱动后，RaspBian 上配置无线网卡很简单。同样还是使用 vim 修改/etc/network/interfaces。以下是最后修改好的 interfaces 代码：

```
####    回环地址
auto lo
iface lo inet loopback

####    第一个有线网卡
auto eth0
allow-hotplug eth0
#iface eth0 inet manual
iface eth0 inet static
address 192.168.2.91
netmask 255.255.255.0
gateway 192.168.2.1

####    第一个无线网卡
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.2.92
netmask 255.255.255.0
gateway 192.168.2.1
wpa-ssid yourssid
wpa-psk youpassword
```

重启系统，执行命令：

```
sudo reboot
```

好了，系统重启后无线网卡将自动连接到 wifi 上了。可以拔下有线网卡上的网线了。无线连接网络优势在于使用方便，有线连接的优势在于性能优越。根据需要自行选择。



注 意

不管是选择有线连接还是无线连接，建议都是用静态分配 IP。这是为了以后 Putty 连接方便。总不能每次连接 Raspberry 前都扫描一次内网确定 Raspberry 的 IP 吧。

2.2.4 其他配置

/etc/hosts 文件保存的是与 IP 对应的主机名。在 Raspberry 解析网络域名时，它首先就是查询 /etc/hosts 文件，是否有这个域名存在。如果有，则不查询 DNS，直接在本地解析域名的 IP。如果没有，则向上一级查询域名。

执行命令：

```
nano -w /etc/hosts
```

在文件末尾添加“192.168.2.91　　pi”，按 Ctrl+x 组合键保存文件，输入 Y，确认保存。这里不需要重启服务直接生效。



注 意

如果在这个文件末尾添加了“192.168.2.91　　www.baidu.com”会怎么样呢？没错，当你用 Raspberry 浏览百度网页的时候，实际指向的却是 Raspberry 的 web 服务。

/etc/resolv.conf，这个文件保存的是 DNS 信息。执行命令 ：

```
nano -w /etc/resolv.conf
```

在百度里搜索一下网络供应商提供的 DNS，例如武汉电信的 DNS 就是 202.103.24.68。修改文件内容为“nameserver 202.103.24.68”，按 Ctrl+x 组合键保存文件，输入 Y，确认保存。这里不需要重启服务。

下面开始修改 Raspberry 的更新源，这是最为重要的修改。Raspberry 的更新源实际就是一个个服务器的地址。Raspberry 默认的更新源都在国外。我们用 apt-get 安装软件，更新系统速度比较慢。因此，我们要把 Raspberry 的更新源换成国内的。/etc/apt/sources.list，这个文件保存的就是更新源的信息。

到 RaspBian 的官网查看一下更新源的镜像信息。使用浏览器打开 http://www.raspbian.org/RaspbianMirrors。在网页上按 Ctrl + f 组合键 ，查找 China，如图 2-21 所示。

Asia	China	Tsinghua University Network Administrators	http://mirrors.tuna.tsinghua.edu.cn/raspbian/raspbian/
Asia	China	Dalian Neusoft University of Information	http://mirrors.neusoft.edu.cn/raspbian/raspbian
Asia	China	Cohesion Network Security Studio (CNSS)	http://raspbian.cnssuestc.org/raspbian/ rsync://raspbian.cnssuestc.org/raspbian
Asia	China	Unique Studio of Huazhong University of Science and Technology	(http|rsync)://mirrors.hustunique.com/raspbian/raspbian
Asia	China	University of Science and Technology of China	(http|rsync)://mirrors.ustc.edu.cn/raspbian/raspbian/
Asia	China	SUN YAT-SEN University	http://mirror.sysu.edu.cn/raspbian/
Asia*	China	Zhejiang University	http://mirrors.zju.edu.cn/raspbian/raspbian/
Asia*	China	Open Source Software Association of Chinese Academy of Sciences	http://mirrors.opencas.cn/raspbian/raspbian/
Asia	China	Chongqing University	http://mirrors.cqu.edu.cn/Raspbian/raspbian/

图 2-21　RaspBian 更新源

以上都是 RaspBian 推荐的官方源，实际上还有其他的源可用，比如：

- 中国科学技术大学：RaspBian http://mirrors.ustc.edu.cn/raspbian/raspbian/
- 搜狐源：RaspBian http://mirrors.sohu.com/raspbian/raspbian/

但还是官方推荐源用得比较放心。执行命令：

```
nano -w /etc/apt/sources.list
```

直接先把官方源去掉或者前面加#号注释掉，添入以下源：

```
deb http://mirrors.ustc.edu.cn/raspbian/raspbian/ wheezy main contrib non-free
 rpi
```

按 Ctrl + x 组合键保存文件，输入 Y，确认保存。无须重启服务，设置直接生效，只需要执行命令：

```
sudo apt-get update
sudo apt-get upgrade
```

第一条命令的作用是更新源列表，第二条命令的作用是更新系统。


注 意

到了这一步，Raspberry 基本上已经配置完毕可以使用了。后面的步骤只是为了更方便地使用。

2.3　远程无密码登录

大概很少有人专门为 Raspberry 配块显示器吧。大多数人都是使用远程登录软件，用 ssh 来连接 Raspberry 的。但麻烦的是每次登录都得一次次地输入密码。为了使用方便，可以为 Raspberry

创建一套钥匙。本章将简单说明公钥私钥，使用 ssh 工具远程无密码登录 Raspberry。

2.3.1　Windows 远程无密码登录

Putty 是一个 Telnet、SSH、Rlogin、纯 TCP 以及串行接口连接软件，它支持 Windows、Linux、Bsd 平台，据说正在开发 Mac OS 版本的 Putty。Putty 体积小、功能强，使用方便，是 SSH 连接工具中的明星软件。本书统一使用 Putty 来连接 Raspberry。

1. 确定网络

首先得确定，Raspberry 和正在使用的 Windows PC 在同一局域网内，或者两者之间能连通。单击“开始”→“附件”→“命令提示符”打开命令窗口，如图 2-22 所示。

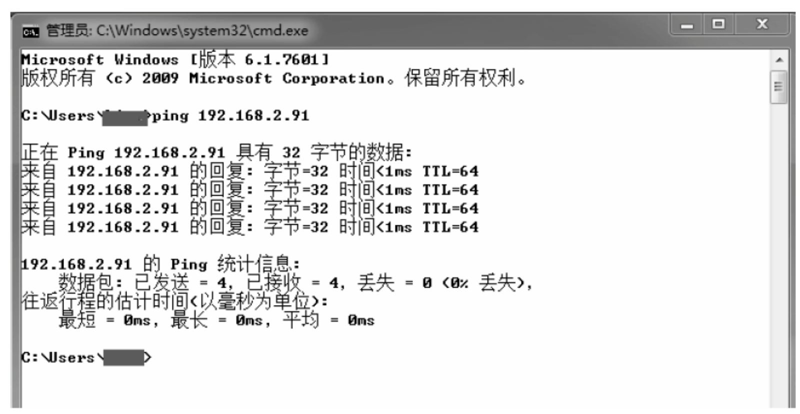


图 2-22　测试网络

2. 登录 Raspberry

刚配置好的 Raspberry 的 IP 设置的是 192.168.2.91。我们先打开 Putty，如图 2-23。

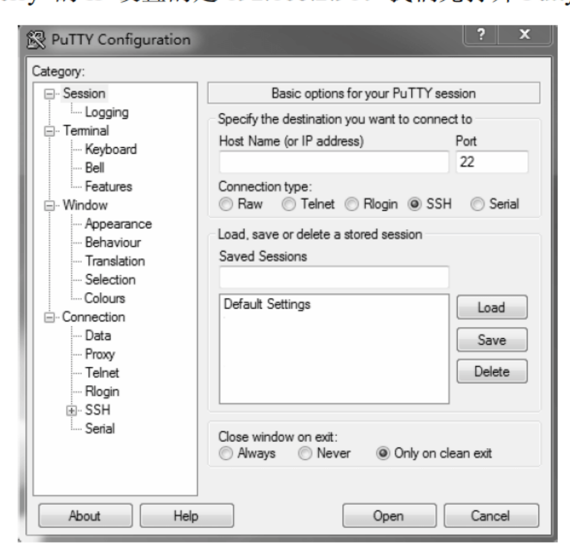


图 2-23　Putty

在 Host Name （or IP address）下面的文本框中输入 Raspberry 的 IP 地址，按 Save 按钮，创建了一个 Putty 的会话（session），如图 2-24 所示。

```
login as:
```

图 2-24 输入用户名

输入默认的用户名 pi，按 Enter 键，界面如图 2-25 所示。

```
login as: pi
pi@192.168.2.1'' password:
```

图 2-25 输入密码

输入配置 Raspberry 时设定的密码后再按 Enter 键。现在就登录到了 Raspberry 上了。怎样做到无密码登录呢？这里就先得说说 Public Key（公钥）和 Private Key（私钥）了。我们暂时可以简单地理解成锁和钥匙的关系。公钥是锁，锁住 Raspberry（用在服务端）；私钥是钥匙，用来开锁登录（用在客户端）。实际上当然没这么简单，但在这里，我们也只需要有这个概念就可以了。更复杂详细的解释，请参考百度。

在 Raspberry 上，使用 ssh-keygen 命令来生成公钥和私钥。先来 man 一下 ssh-kengen 命令，如图 2-26 所示。

```
SSH-KEYGEN(1)              BSD General Commands Manual              SSH-KEYGEN(1)

NAME
     ssh-keygen — authentication key generation, management and conversion

SYNOPSIS
     ssh-keygen [-q] [-b bits] -t type [-N new passphrase] [-C comment]
                [-f output keyfile]
     ssh-keygen -p [-P old passphrase] [-N new passphrase] [-f keyfile]
     ssh-keygen -i [-m key format] [-f input keyfile]
     ssh-keygen -e [-m key format] [-f input keyfile]
     ssh-keygen -y [-f input keyfile]
     ssh-keygen -c [-P passphrase] [-C comment] [-f keyfile]
     ssh-keygen -l [-f input keyfile]
     ssh-keygen -B [-f input keyfile]
     ssh-keygen -D pkcs11
     ssh-keygen -F hostname [-f known hosts file] [-l]
     ssh-keygen -H [-f known hosts file]
     ssh-keygen -R hostname [-f known hosts file]
     ssh-keygen -r hostname [-f input keyfile] [-g]
     ssh-keygen -G output file [-v] [-b bits] [-M memory] [-S start point]
     ssh-keygen -T output file -f input file [-v] [-a num trials] [-K checkpt]
                [-W generator]
```

图 2-26 man ssh-keyen

ssh-kengen 的参数很多，我们只需要知道其中的两个就可以了。

- -P：提供密码。
- -t：加密方式，可以使用：rsa1（SSH-1）rsa（SSH-2）dsa（SSH-2）。

3. 创建公钥、私钥

在刚登录的 Putty 会话中输入命令：

```
ssh-keygen -t rsa -P ""
```



注 意

这里的命令不需要加 sudo，直接登录用户执行命令。

这个命令的作用是，使用 ssh-keygen 命令来创建一对密钥，加密方式是 rsa，密码为空。按 Enter 键，就会在/home/pi/目录下创建了.ssh 目录和.ssh/id_rsa、.ssh/id_rsa.pub 文件。其中 id_rsa 就是私钥，id_rsa_pub 就是公钥。

4. 公钥作用于服务端

现在我们把锁（公钥 public key）挂到 Raspberry 的大门上。执行命令：

```
cat /home/pi/.ssh/id_rsa.pub >> /home/pi/.ssh/authorized_keys
```

5. 私钥传至客户端

把私钥分发给需要登录 Raspberry 的主机就可以了。

下面使用 WinSCP 这个软件，将 Raspberry 上的私钥 id_rsa 拷贝到 Windows 下，如图 2-27 所示。

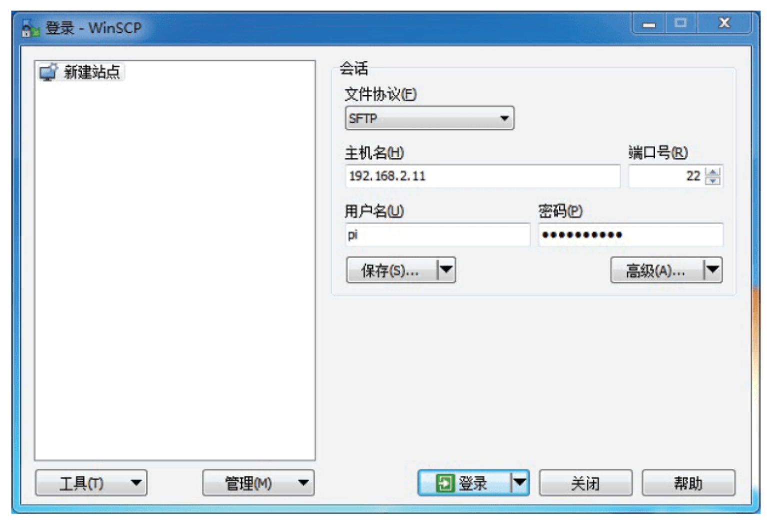


图 2-27　WinSCP

在主机名下面的文本框输入 Raspberry 的 IP，用户名下面的文本框输入 pi，密码下面的文本框输入 Raspberry 的密码。单击“登录”按钮，出现如图 2-28 所示窗口。

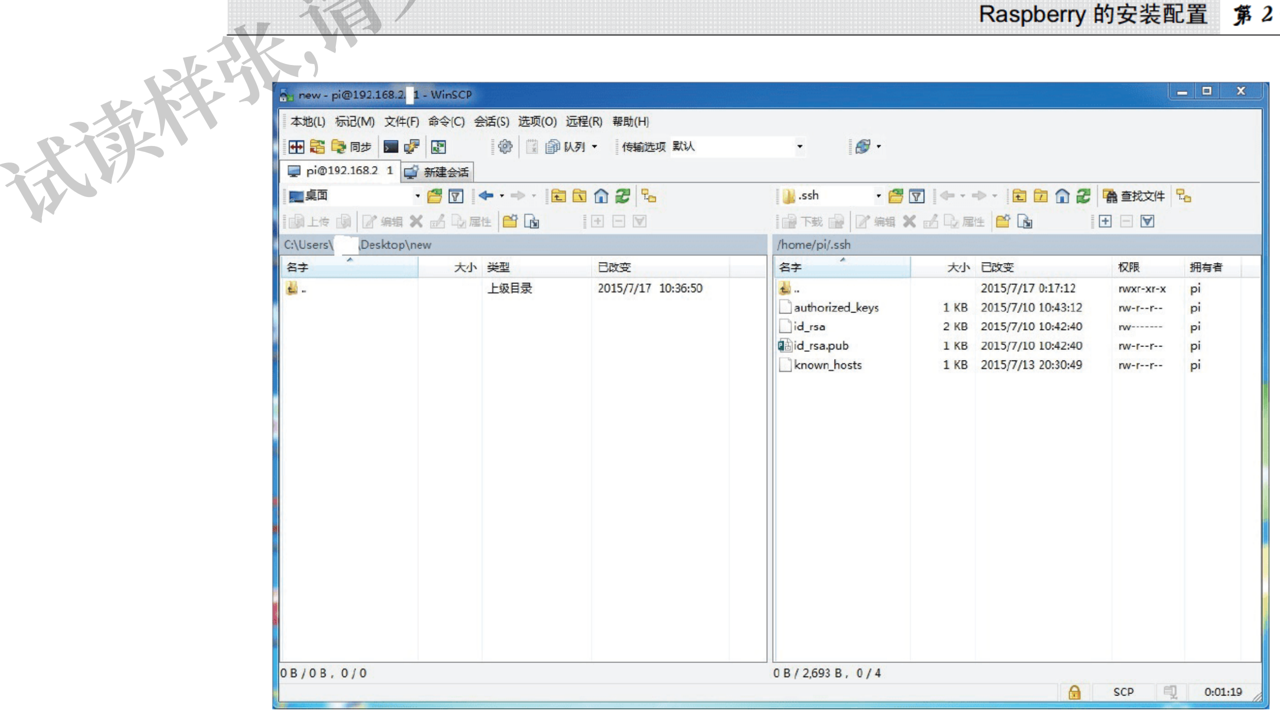


图 2-28 winscp 登录

左边窗口显示的是 Windows 目录，右边窗口显示的是 Raspberry 的目录。单击 id_rsa，将其拖动到左边的目录下。好了，现在 id_rsa 私钥就被传到 Windows 目录下了。

6. 转换私钥

Putty 并不能直接使用这个私钥，我们还得经过一道手续才行。先在 Putty 目录创建一个 keys 文件夹，这个文件夹建在哪里都一样，放到 Putty 目录下只是为了方便而已。假设目录为 c:\putty\。打开 Putty 目录下的 puttygen.exe，如图 2-29 所示。



图 2-29 puttygen

单击 File 菜单中的 load private key 选项，打开的对话框如图 2-30 所示。

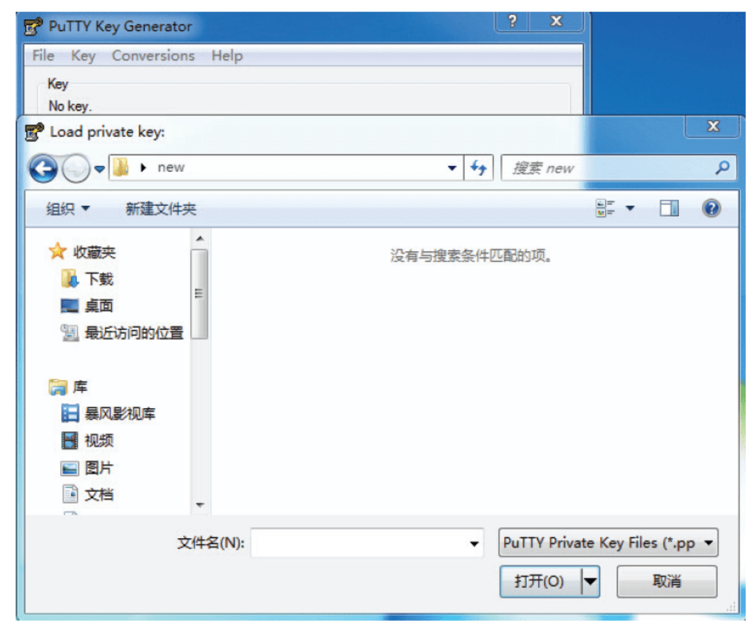


图 2-30 Load private key

怎么什么都没有？没关系，单击“取消”按钮上面的下拉框，将其选取成 All Files（*.*），现在 id_rsa 显示出来了。单击 id_rsa 私钥文件，再单击“打开”按钮，如图 2-31 所示。

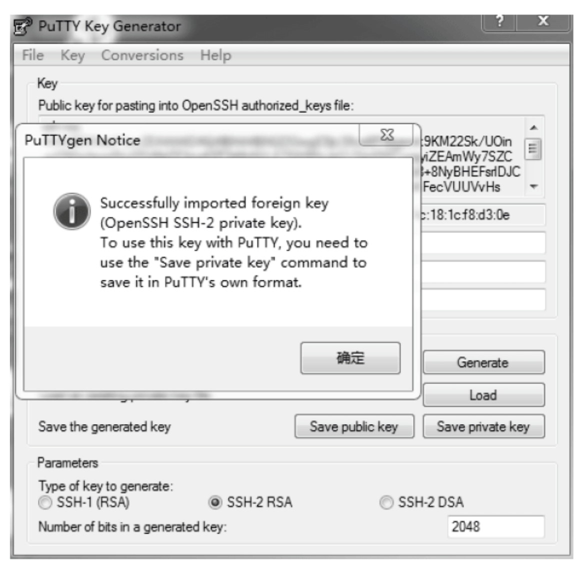


图 2-31 Select private key

单击“确定”按钮，然后单击 Save private key 按钮，如图 2-32 所示。

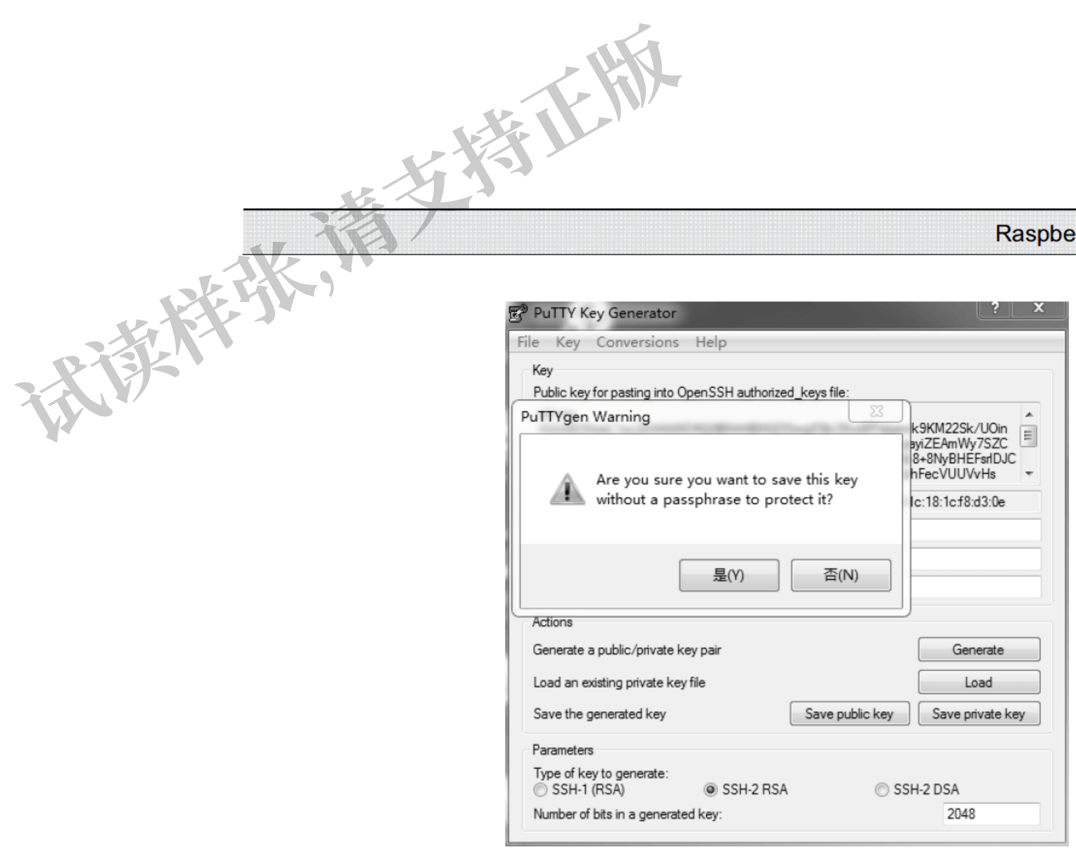


图 2-32　Save private key

单击“是（Y）”按钮，出现如图 2-33 所示界面。

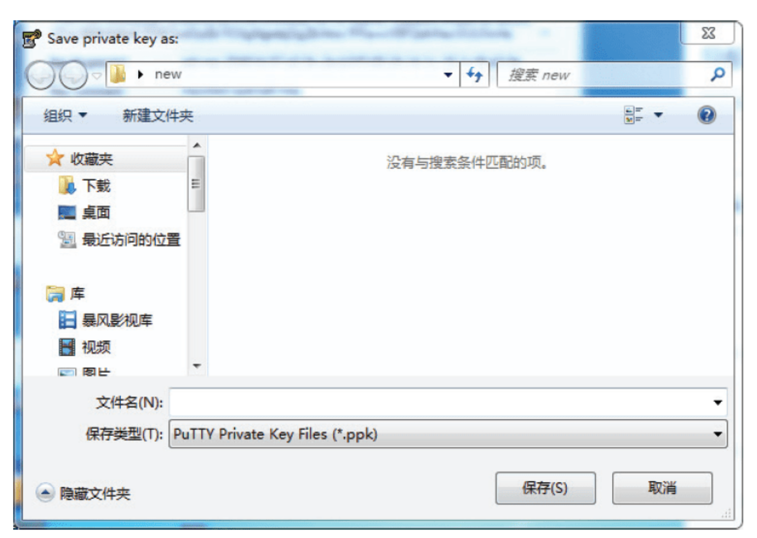


图 2-33　保存 Putty 密钥

在文件名后面的文本框中输入密钥名 pi，单击“保存”按钮，得到了 pi.ppk 文件。将 pi.ppk 文件拷贝到 Putty 目录中刚创建的 keys 目录下。

7. 创建快捷方式

在 Windows 桌面的空白处，右击打开桌面菜单。单击“新建”→“快捷方式”选项，打开的对话框如图 2-34 所示。

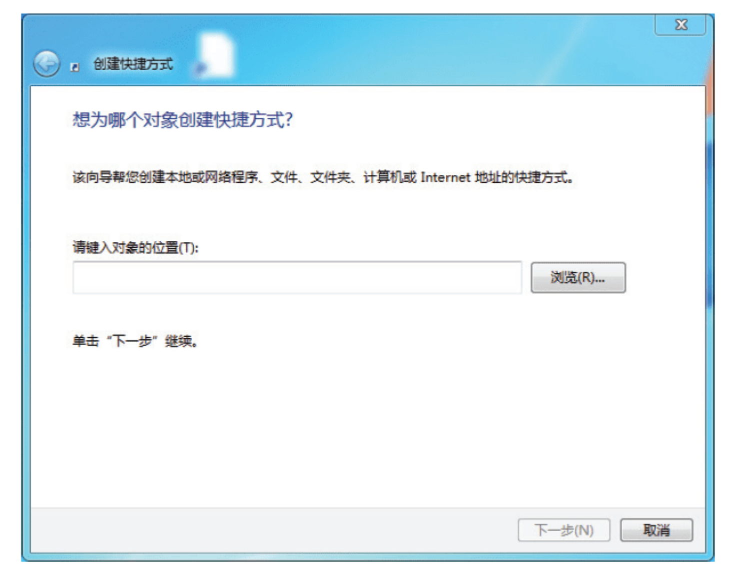


图 2-34　创建快捷方式

在“请键入对象的位置”下面的文本框输入：

```
"c:\PUTTY.EXE" -i "c:\putty\key\pi.ppk " pi@192.168.2.91
```



注　意

这里是假设 Putty 的目录是 c:\putty，可根据自己的实际情况修改。

单击“下一步”按钮，出现图 2-35 所示界面。

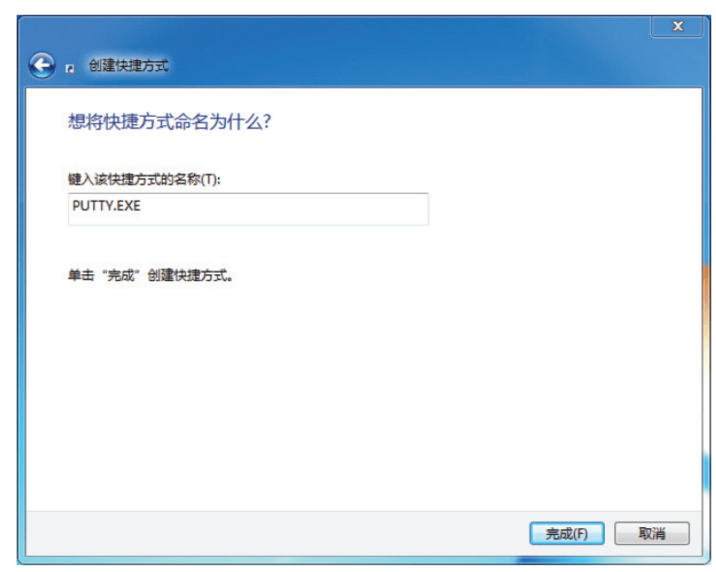


图 2-35　快捷方式完成

在“键入该快捷方式的名称”下的文本框输入 Raspberry，单击“完成”按钮，就得到了一个名为 Raspberry 的 Putty 快捷方式。现在完成了，双击 Raspberry 快捷方式，就可以直接登录 Raspberry 了，如图 2-36 所示。如果想漂亮一点，可自行下载一个 Rasbperry 的 icon 图标，换到这个快捷方式上。

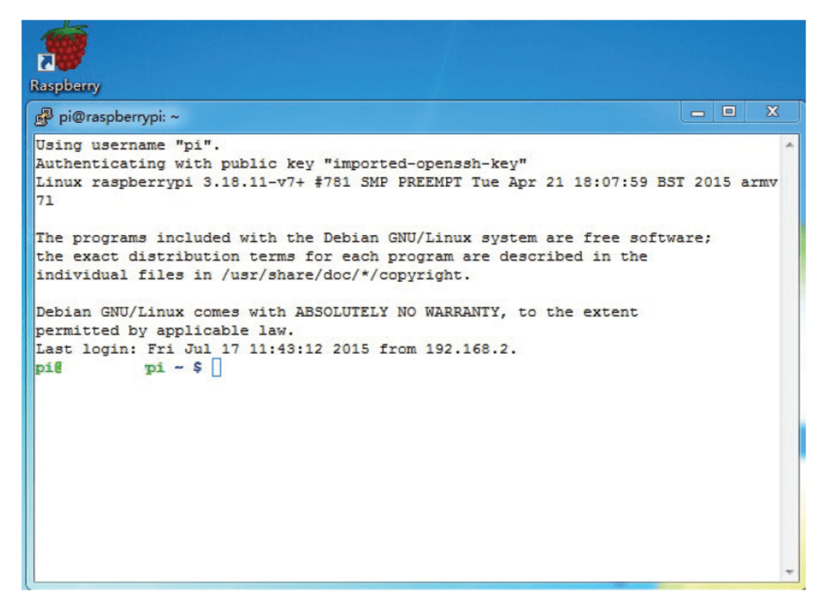


图 2-36　Putty 无密码登录

2.3.2　Linux 远程无密码登录

在 Linux 上用 ssh 无密码登录，与 Windows 上的原理是一样的。Raspberry 也属于 Linux，师出同门就更方便了。

1. 登录客户端

在 Linux 下，打开 Terminal。或者直接按 Ctrl + Alt + F2 组合键，如图 2-37 所示进入控制台。

```
Debian GUN/Linux 7 debian7 tty2

debian7 login:
```

图 2-37　Linux 控制台

2. 在客户端创建公钥、私钥

输入用户名密码登录后。这里不需要用 root 登录，一般用户都可以。执行命令：

```
ssh-keygen -t rsa -P ""
```

按 Enter 键就会在登录用户的家目录下创建.ssh 目录和.ssh/id_rsa、.ssh/id_rsa.pub 文件。

3. 将公钥传至服务端（Raspberry）

下面我们要将 Linux 中创建的公钥 id_rsa.pub 传输到 Raspberry。执行命令：

```
ssh pi@192.168.2.91
```

输入 pi 的密码，登录到了 Raspberry，如图 2-38 所示。

```
Last login: Sat Jul 18 01:48:10 2015 from debian.local
pi@raspberrypi ~ $ 
```

图 2-38　登录到 Raspberry

执行命令：

```
scp                          loginName@LinuxIP:/home/loginName/.ssh/id_rsa.pub
/home/pi/.ssh/linux.pubkey
```

这里需要输入 Linux 用户的密码。

4. 公钥作用于服务端

把锁挂在大门上，运行命令：

```
cd .ssh
cat linux.pubkey >> authorized_keys
```



loginName 是 Linux 用户的登录名。LinuxIP 是 Linux 主机的 IP。

好了，到这一步就可以了。执行命令：

```
exit
```

现在又回到了 Linux。再次执行命令：

```
ssh pi@192.168.2.91
```

这里就不再需要输入密码了，可以直接登录到 Raspberry 上了。



在 Windows 登录时使用的是服务端（Raspberry）创建公钥、私钥。在 Linux 登录时使用的是客户端（Linux）创建公钥、私钥。这两种方式效果是一样的。

2.4　系统备份和还原

好不容易安装配置好 RaspBian 系统，如果因为一个小失误却不得不重装系统，然后再配置一遍。费时费力，那真是让人抓狂。如果在 Windows 下该怎么办呢？很简单用 ghost 备份。Linux 下同样简单，还是备份。本章将熟悉备份、还原工具，熟练掌握备份、还原系统。

Linux 的备份有很多方法，这里我们来讲解其中最简单的两种：一种是以 tar 来压缩，一种是以 dd 命令来备份。

2.4.1 tar 备份还原

1. tar 备份系统

首先来看下 tar 命令的作用，如图 2-39 所示。

```
pi@raspberrypi: ~
TAR(1)                    BSD General Commands Manual                    TAR(1)

NAME
     tar — The GNU version of the tar archiving utility

SYNOPSIS
     tar [-] A --catenate --concatenate | c --create | d --diff --compare |
         --delete | r --append | t --list | --test-label | u --update | x
         --extract --get [options] [pathname ...]

DESCRIPTION
     Tar stores and extracts files from a tape or disk archive.

     The first argument to tar should be a function; either one of the letters
     Acdrtux, or one of the long function names.  A function letter need not
     be prefixed with ``-'', and may be combined with other single-letter
     options.  A long function name must be prefixed with --.  Some options
     take a parameter; with the single-letter form these must be given as sep‐
     arate arguments.  With the long form, they may be given by appending
     =value to the option.

FUNCTION LETTERS
     Main operation mode:
 Manual page tar(1) line 1 (press h for help or q to quit)
```

图 2-39　man tar

tar 是一个打包程序。有点类似于 Windows 下的 Winrar。但它没有压缩功能，如果需要压缩，还得配合 gzip 一起使用。

tar 的参数有很多。常用的几个参数如下。

- -c：建立一个压缩文件的参数指令（create 的意思）。
- -x：解开一个压缩文件的参数指令。
- -t：查看 tarfile 里面的文件。



注 意

在参数的下达中，c/x/t 仅能存在一个！不可同时存在！因为不可能同时压缩与解压缩。

- -z: gzip 压缩/解压缩。
- -j: bzip2 压缩/解压缩。
- -v: 压缩的过程中显示文件。
- -f: 使用文件名，请留意，在 f 之后要立即接文件名，不要再加参数。例如使用 tar -zcvfP tfile sfile 就是错误的写法，要写成 tar -zcvPf tfile sfile 才对。
- -g: 增量备份
- -p: 保留原文件的原来属性。
- -P: 可以使用绝对路径来压缩。

- -N：比后面接的日期（yyyy/mm/dd）还要新的才会被打包进新建的文件中。
- --exclude FILE：在压缩的过程中，不要将 FILE 打包。

好了，下面正式开始备份步骤。

（1）查看需要备份的目录

使用 Putty 登录 Raspberry 后执行命令：

```
ls /
```

查看 Raspberry 的根目录，如图 2-40 所示。

```
pi@raspberrypi ~ $ ls /
bin   dev  home  lost+found  mnt  proc  run   selinux  sys  usr
boot  etc  lib   media       opt  root  sbin  srv      tmp  var
pi@raspberrypi ~ $ 
```

图 2-40　根目录

并不是整个系统都需要备份的，有些目录完全可以略过。

- lost+found：存放修复或损坏的文件的目录，一般情况下里面没有东西。
- mnt：一般用来挂载硬盘优盘的目录。
- proc：目录文件，只存在内存当中，而不占用外存空间。
- sys：内核信息映射。
- media：一般用来挂载光盘。
- tmp：临时文件。

也就是说以上的几个目录是可以不打包的。

（2）进入备份目录，开始备份

原理弄清楚了，下面执行命令：

```
cd /tmp
tar zcvpf pi_20150718.tar.gz -exclude=/lost+found -exclude=/mnt -exclude=/sys
-exclude=/proc -exclude=/media -exclude=/tmp /
```

第一条命令是进入/tmp 备份目录下。第二条命令作用是，除了以上几个文件夹外，使用 gzip 压缩，打包整个系统。压缩文件名为 pi_20150718.tar.gz。

等命令执行完毕后，找个大容量的优盘挂载到 pi 上，将 pi_20150718.tar.gz 转移到优盘保存，或者利用 scp 命令将该备份文件转移到其他 PC 上。



注　意

使用 tar 备份，可以直接在 Raspberry 上执行。也就是说在本机来备份本机，有点类似于 GHOST 的备份。刚才的例子里，我是在/tmp 目录下进行备份的，优点就是速度快，毕竟只需要在硬盘上读取。缺点是在这里备份的前提条件是/tmp 目录下有足够的空间。如果没有，那就找块大容量的移动硬盘或者是优盘，把它挂载到/mnt 目录上，再进入/mnt 目录来备份整个系统。这样就涉及 USB 的传输速度什么的了，速度就差了一点点。

2. tar 还原系统

tar 还原就简单多了。将备份文件 pi_20150718.tar.gz 拷贝到/tmp 下，执行命令：

```
tar zxvpf pi_20150718.tar.gz -C /
```

好了，现在系统已经恢复到备份时一样了。

2.4.2　tar 增量备份还原

tar 备份虽好，但是每次备份都得全系统备份未免太浪费空间了。别急，tar 还有一个增量备份。除了第一次是备份整个系统外，以后每次备份都只备份比上次多出来的部分。这样就无所谓浪费空间了。

1. tar 增量备份系统

tar 的增量备份实际上就是利用了 tar 的-g 参数。第一次还是先备份系统，或者说是基础备份，后面的增量备份都是以这个备份文件为基准的。执行命令：

```
cd /tmp
sudo tar -g snapshot -zcvpf pi.tar.gz / -exclude=/lost+found -exclude=/mnt
-exclude=/sys -exclude=/proc -exclude=/media -exclude=/tmp
```

第二次就可以进行增量备份了。执行命令：

```
cd /tmp
sudo tar -g snapshot -zcvpf pi_incremental_1.tar.gz / -exclude=/lost+found
-exclude=/mnt -exclude=/sys -exclude=/proc -exclude=/media -exclude=/tmp
```

第 3 次、第 4 次……同样处理。只要将 incremental_后面的数字递增就可以了。通过 ls –al 命令查看递增备份文件的 ctime。在还原系统时，甚至可以做到定点还原。


注 意

在创建递增备份文件时，文件名必须是以基础备份文件名开头，后面紧跟着_incremental_，再后面跟着增量备份的次数。

2. tar 增量还原系统

先查看几个增量备份文件的 ctime。根据需要定点还原系统，如图 2-41 所示。

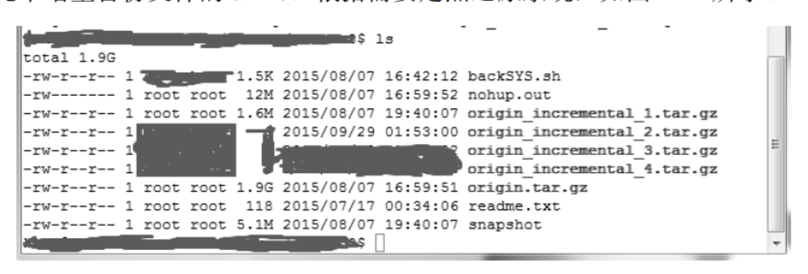


图 2-41　备份系统

图 2-41 是系统的增量备份文件。如果想将系统恢复到 2015/08/07，图中只有 orgin.tar.gz 和 orgin_incremental_1.tar.gz 的 ctime 是 2015/08/07 的。那就执行命令：

```
sudo tar zxvf orgin.tar.gz -C /
sudo tar zxvf orgin_incremental_1.tar.gz -C /
```

如果想恢复到 2015/09/29，那就再将 ctime 为 2015/09/29 的压缩文件解压缩到根目录下就可以了。

2.4.3 dd 备份还原

用 dd 命令来备份还原系统，最简单的方法是硬盘对拷。这是最简单粗暴的备份还原方法了。如果用两个容量一样的硬盘对拷，可以拷贝出完全相同的 2 份系统。

1. dd 备份系统

Ghost 是 Windows 下备份系统的明星软件。它主要作用是将系统分区打包成*.gho 文件备份。或者将*.gho 文件还原到分区。dd 命令的作用跟它很类似。至于 dd 命令，我们在安装系统到 TF 卡的时候已经用过了。既然它可以把一个*.img 文件写入磁盘。当然也可以从磁盘备份文件到*.img 文件。下面的例子是以 Debian 7 做操作平台。

（1）登录系统

先进入 Linux 的控制台，参考 2.1.3 小节，如图 2-37 所示，使用 root 用户登录。

（2）确定备份位置

将 Raspberry 的 TF 卡卸下来，装入读卡器中，插入到 Linux 主机的 USB 接口。使用 ls –l /dev/sd* 命令查看 TF 卡磁盘位置，如图 2-42 所示。

```
ls -l /dev/sd*
```

```
root@debian7:~# ls -l /dev/sd*
brw-rw---T 1 root disk      8,   0 Jun 28 18:07 /dev/sda
brw-rw---T 1 root disk      8,   1 Jun 28 18:07 /dev/sda1
brw-rw---T 1 root disk      8,   2 Jun 28 18:07 /dev/sda2
brw-rw---T 1 root disk      8,   5 Jun 28 18:07 /dev/sda5
brw-rw---T 1 root floppy    8, 16 Jun 28 18:28 /dev/sdb
brw-rw---T 1 root floppy    8, 17 Jun 28 18:28 /dev/sd1
root@debian7:~# umount /dev/sdb1
umount: /dev/sdb1: not mounted
root@debian7:~# _
```

图 2-42　查看 TF 卡

（3）备份系统

备份有两种方法：直接 dd 备份系统、dd 压缩备份系统。

- 直接 dd 备份系统。执行命令:

```
cd
dd if=/dev/sdb of=Raspberry_20150719.img
```

第一条命令是进入 root 用户的家目录。第二条命令是将/dev/sdb 磁盘写入到 Raspberry_20150719.img 文件中。第二条命令是不是很眼熟？没错，就是我们安装系统时的命令反过来用。

但是这样备份有个弊端。就是 sdb 磁盘有多大，备份用的 img 文件就有多大。那安装的时候为什么 1G+的 img 系统怎么安装到了大磁盘中的呢？还记得 Raspberry 的配置选项中有个扩展分区的选项吗？实际上在我们安装系统的时候，也只是用了 1GB 左右的空间，后来系统占满整块硬盘，是因为我们配置 Raspberry 时扩展了分区。

dd 备份系统与 tar 备份系统不一样，tar 备份的时候可以选择哪些文件夹不备份，dd 是不能选择的，它只能整体备份还原。

- dd 压缩备份系统

为了将整块磁盘备份到一个比较小的文件。我们用 gzip 将它压缩一下再备份，这样就好多了。执行命令：

```
dd if=/dev/sdb | gzip -9 > Raspberry_20150719.img.gz
```

2. dd 还原系统

针对两种备份形式，还原也分两种：dd 直接还原系统、dd 解压缩还原系统。

- dd 直接还原系统，执行命令:

```
dd  if=Raspberry_20150719.img of=/dev/sdb
```

跟安装是一模一样的。

- dd 解压缩还原系统，执行命令:

```
gzip -c -d Raspberry_20150719.img.gz | dd of=/dev/sdb
```

好了，现在再也不用担心我的系统了，开始放心大胆地折腾吧。

第 3 章 Raspberry开发利器

工欲善其事，必先利其器。操作系统安装完毕，这只是长征走完第一步。光有系统没有工具的配合，Raspberry 能做的事非常有限。下面开始安装 Linux 工具软件，熟练地使用工具后才能进行其他的项目。

Linux 中的软件程序很多，但常用的就这么几种：bash、python、vim、nano、sed、awk、grep、find……它们大多都是命令行工具，学习起来可能稍稍困难，但学会了以后的工作就非常简单了。可以说，只要熟练地掌握了这几种工具，应用 Linux 就基本没什么问题了。

介绍这个工具之前，得先学习另一个命令 apt-get。只有熟悉了它，才能把程序软件安装到系统上，上章在更换更新源时已经使用过了。

本章主要内容包括：

- 掌握 apt-get 命令
- 学会 vim 的基本操作
- 学习 bash
- 简单学习 Python 脚本语言

3.1 apt-get

在 Linux 中安装软件一般有两种方式：

一种是源代码安装，这种方法的好处是能自行调整参数，特别适合自己的机器（gentoo 就是所有的软件都会被重新编译安装，因此 gentoo 也是所有 Linux 发行版本中最适合自己机器的版本）。但编译起来比较复杂，而且以现在的机器配置来说似乎也没多大的必要。如果不是非得编译安装的软件，还是用简单的方法吧。

另一种就是用包管理器来安装，比如 Debian 的 apt-get 和 RedHat 的 yum（目前最新的包管理器是 dnf）。这种安装方法简单方便，而且所有的打包软件都经过了发行版本官方的验证，可以保证安全。目前绝大部分的程序软件都可以用这种方法安装，适合初学者使用。

3.1.1 apt-get 简介

Advanced Package Tool，又名 apt-get，是一款适用于 UNIX 和 Linux 系统的应用程序管理器。最初于 1998 年发布，用于检索应用程序并将其加载到 Debian Linux 系统。apt-get 成名的原因之一

在于其出色的解决软件依赖关系的能力。它通常使用.deb-formatted 文件，但经过修改后可以使用 apt-rpm 处理红帽的 Package Manager（RPM）文件。

Apt-get 在 Linux 社区得到广泛使用，成为用来管理桌面、笔记本和网络的重要工具。使用 apt-get 的主流 Linux 系统包括 Debian 和 Ubuntu 变异版本。大多数情况下，从命令行运行该工具。

3.1.2 apt 命令用法

想要了解某个命令很简单，使用 man command 就行了。下面来看一下 man apt-get，如图 3-1 所示。

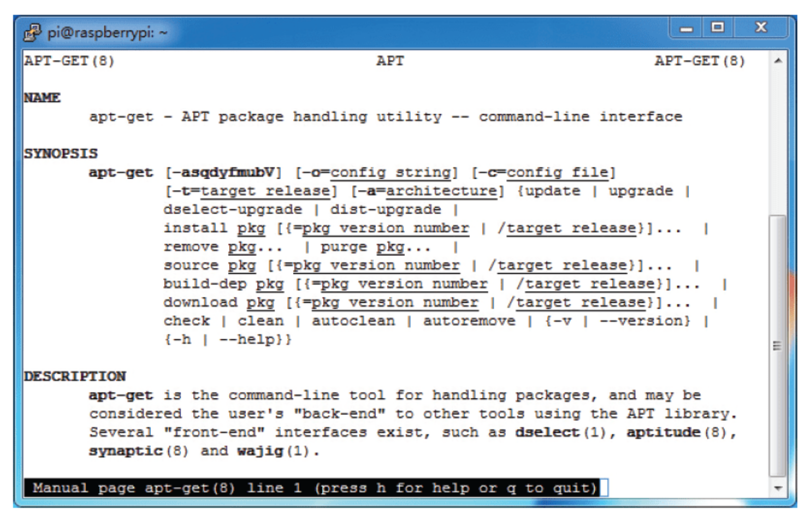


图 3-1　man apt-get

Packagename 是软件包的名称，具体的命令如下：

- apt-get update：在修改/etc/apt/sources.list 之后运行该命令。此外需要定期运行这一命令以确保您的软件包列表是最新的。
- apt-get install packagename：安装一个新软件包（参见下文的 aptitude）。
- apt-get remove packagename：卸载一个已安装的软件包（保留配置文档）。
- apt-get remove --purge packagename：卸载一个已安装的软件包（删除配置文档）。
- apt-get autoremove packagename：删除包及其依赖的软件包。
- apt-get autoremove --purge packagname：删除包及其依赖的软件包+配置文件，比上面的要删除的彻底一点。
- dpkg --force-all --purge packagename：有些软件很难卸载，而且还阻止了别的软件的应用，就能够用这个，但是有点冒险。
- apt-get autoclean：apt 会把已装或已卸的软件都备份在硬盘上，所以假如需要空间的话，能够用这个命令来删除已卸载掉的软件的备份。
- apt-get clean：这个命令会把安装的软件的备份也删除，但是不会影响软件的使用。
- apt-get upgrade：可以使用这个命令更新软件包，apt-get upgrade 不仅可以从相同版本号

的发布版中更新软件包，也可以从新版本号的发布版中更新软件包，尽管实现后一种更新的推荐命令为 apt-get dist-upgrade。在运行 apt-get upgrade 命令时加上-u 选项很有用(即 apt-get -u upgrade)。这个选项让 APT 显示完整的可更新软件包列表。不加这个选项，就只能盲目地更新。APT 会下载每个软件包的最新更新版本，然后以合理的次序安装它们。


注 意

在运行该命令前应先运行 apt-get update 更新数据库，更新任何已安装的软件包。

- apt-get dist-upgrade：将系统升级到新版本。
- apt-cache search string：在软件包列表中搜索字符串。dpkg -l package-name-pattern 列出任何和模式相匹配的软件包。假如不知道软件包的全名，可以使用 *package-name-pattern*。
- aptitude：详细查看已安装或可用的软件包。和 apt-get 类似，aptitude 能够通过命令行方式调用，但仅限于某些命令——最常见的有安装和卸载命令。由于 aptitude 比 apt-get 包含更多信息，它更适合用来进行安装和卸载。
- apt-cache showpkg pkgs：显示软件包信息。
- apt-cache dumpavail：打印可用软件包列表。
- apt-cache show pkgs：显示软件包记录，类似于 dpkg –print-avail。
- apt-cache pkgnames：打印软件包列表中任何软件包的名称。
- dpkg -S file：这个文档属于已安装软件包。
- dpkg -L package：列出软件包中的任何文档。
- dpkg –l：列出所有已安装的软件包。
- apt-get autoclean：定期运行这个命令来清除那些已卸载的软件包的.deb 文档。通过这种方式，能够释放大量的磁盘空间。假如需求十分迫切，可以使用 apt-get clean 以释放更多空间。这个命令会将已安装软件包裹的.deb 文档一并删除。大多数情况下不会再用到这些.debs 文档，因此假如我们为磁盘空间不足而感到焦头烂额，这个办法也许值得一试。

apt-get 命令使用方便，功能强大，是学习 Linux 中必须掌握的命令之一(和 rh 系列中的 yum/dnf 作用相同)。虽然现在已经有很多图形化的包管理工具，在我看来它们唯一的优点就是直观。如果熟悉了 apt-get 命令，相信你会做出正确的选择。


注 意

几乎所有的非商业软件都是可以用 apt-get 安装的，即使不能用 apt-get 安装的商业软件，大部分也有 deb 安装包。

3.2 vim

vim 是 Linux 世界中最流行的几种文本编辑器之一，是 Linux 程序员的挚爱，据说很多 Linux

技术大牛编程时不用 IDE 只使用 vim。在绝大多数发行版本中它都是默认安装的。很多场合它不仅仅作为文本编辑器，还被配置成一款优秀的 IDE。如果说到 Linux 的明星软件，必定有 vim 的一席之地。

3.2.1 vim 简介

vim 是一个类似于 vi 的文本编辑器，不过在 vi 的基础上增加了很多新的特性，是 vi 的加强版。vim 普遍被推崇为类 vi 编辑器中最好的一个。vim 的原意是 visual interface，它是一个所见所得的编辑程序，也就是说可以立即看到操作结果。

3.2.2 安装配置 vim

RaspBian 默认没有安装 vim，只是安装了 vi。下面先来安装 vim。安装 vim 很简单，在 2.3.1 中建立了 Raspberry 无密码登录的 Putty 的快捷方式。右击该图标，Putty 将以默认用户 pi 登录 Raspberry。执行命令：

```
sudo apt-get  install vim
```

执行效果如图 3-2 所示。

```
pi@raspberrypi: ~
Using username "pi".
Authenticating with public key "imported-openssh-key"
Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv
7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Sep 29 16:42:32 2015 from win7.local
pi@raspberrypi ~ $ sudo apt-get install vim
```

图 3-2　安装 vim

Raspberry 上有个默认的 vi，把这个 vi 链接到 vim 上。毕竟现在大家所说的 vi 就是指 vim。如图 3-3 所示，执行命令：

```
whereis vi
whereis vim
ls -l /usr/bin/vi
sudo rm /usr/bin/vi
sudo ln -s /usr/bin/vim /usr/bin/vi
ls -l /usr/bin/vi
```

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ whereis vi
vi: /usr/bin/vi /usr/bin/X11/vi /usr/share/man/man1/vi.1.gz
pi@raspberrypi ~ $ whereis vim
vim: /usr/bin/vim.basic /usr/bin/vim.tiny /usr/bin/vim /etc/vim /usr/bin/X11/vim
.basic /usr/bin/X11/vim.tiny /usr/bin/X11/vim /usr/share/vim /usr/share/man/man1
/vim.1.gz
pi@raspberrypi ~ $ ls -l /usr/bin/vi
lrwxrwxrwx 1 root root 20 Aug 28 12:06 /usr/bin/vi -> /etc/alternatives/vi
pi@raspberrypi ~ $ sudo rm /usr/bin/vi
pi@raspberrypi ~ $ sudo ln -s /usr/bin/vim /usr/bin/vi
pi@raspberrypi ~ $ ls -l /usr/bin/vi
lrwxrwxrwx 1 root root 12 Aug 28 12:07 /usr/bin/vi -> /usr/bin/vim
pi@raspberrypi ~ $ 
```

图 3-3　链接 vim

好了，vim 安装完毕了。现在开始配置 vim。vim 有两个配置文件：一个是全局配置文件/etc/vim/vimrc（有的是/etc/vimrc）；另一个是个人配置文件，在家目录下的.vimrc 文件。此时用 pi 用户登录，那么 pi 用户的 vim 个人配置文件就是/home/pi/.vimrc。这个文件是不可见的，Linux 中文件名的第一个字符是“.”，那么这个文件有点类似于 Windows 中的隐藏文件。

这两个配置文件的优先级是个人配置优先于系统配置。就是说，同一个配置选项，比如/etc/vim/vimrc 中设置的是 set nu，可显示行号。而/home/pi/.vimrc 中设置的是 set nonu，不显示行号。那么在 pi 用户使用 vim 的时候，就不会显示行号，以 pi 用户的配置文件为主。如果没有个人配置文件/home/pi/.vimrc，或者个人配置文件中没有的配置选项，当然是以系统配置为主。如果没有特殊需要，建议修改个人的配置文件比较好。

vim 的配置选项非常多，大概有几百项。以下只列出常用的几项，如表 3-1 所示。如果对此有兴趣可以百度一下。

表 3-1　vim 的配置选项例子

set nocompatible	关闭 vi 兼容模式
syntax on	自动语法高亮
colorscheme molokai	设定配色方案
set number	显示行号
set cursorline	突出显示当前行
set ruler	打开状态栏标尺
set shiftwidth=4	设定 > 命令移动时的宽度为 4
set softtabstop=4	使得按退格键时可以一次删掉 4 个空格
set tabstop=4	设定 tab 长度为 4
set nobackup	覆盖文件时不备份
set autochdir	自动切换当前目录为当前文件所在的目录
filetype plugin indent on	开启插件
set backupcopy=yes	设置备份时的行为为覆盖
set ignorecase smartcase	搜索时忽略大小写，但在有一个或以上大写字母时仍保持对大小写敏感

set nowrapscan	禁止在搜索到文件两端时重新搜索
set incsearch	输入搜索内容时就显示搜索结果
set hlsearch	搜索时高亮显示被找到的文本
set noerrorbells	关闭错误信息响铃
set novisualbell	关闭使用可视响铃代替呼叫
set t_vb=	置空错误铃声的终端代码
set showmatch	插入括号时，短暂地跳转到匹配的对应括号
set matchtime=2	短暂跳转到匹配括号的时间
set magic	设置魔术
set hidden	允许在有未保存的修改时切换缓冲区，此时的修改由 vim 负责保存
set guioptions-=T	隐藏工具栏
set guioptions-=m	隐藏菜单栏
set smartindent	开启新行时使用智能自动缩进
set backspace=indent,eol,start	不设定在插入状态无法用退格键和 Delete 键删除回车符
set cmdheight=1	设定命令行的行数为 1
set laststatus=2	显示状态栏 （默认值为 1，无法显示状态栏）

vim 有丰富的插件，配合 vim 的插件功能，可以将 vim 改造成一个 IDE（for python c/c++ perl ruby ……）。互联网上有详细的教程，按照教程慢慢试验，可配置出最适合自己的 vimrc 文件。


注 意

vim 的赫赫威名在 Linux 界可以自夸一句天下谁人不识君，即使放到 Windows 平台，它也不逊色于 Windows 自带的 notepad（记事本）。

3.2.3 以 vim 做一个简单的 python IDE

既然说到了 IDE，Linux 下的 IDE 非常多，明星软件也不少。但不是来个 IDE 就能满足个人的需求的。为什么非得要人去适应已经固定的 IDE 呢？有了 vim 完全可以配合插件做一个最适合自己的 IDE。下面就以 vim 做一个简单的 python IDE。只添加最简单的功能，如需要扩展其他的功能，请参考资料，自行添加插件。

（1）首先安装 ctags 和 vim 的插件 taglist。

ctags 用于支持 taglist，使用 ctags 可以在变量之间跳跃。执行命令：

```
sudo apt-get install ctags
```

安装 vim 插件 taglist，先安装 vim-scripts，vim-scripts 中带有 vim-addon-manager，vim-addon-manager 是 vim 的插件管理器之一，用来管理 vim 插件。通过 vim-addon-manager 安装 taglist。

```
sudo apt-get install vim-scripts
vim-addons install taglist
```

（2）好了，准备工作已经做好了，现在可以开始设置配置文件了。/etc/vimrc 和~/.vimrc 都可以，只有一个用户，设置哪个都行。如果系统有其他的用户建议还是设置~/.vimrc 比较好。以下是/home/pi/.vimrc 的代码：

```
 1 "文件检测功能
 2 filetype on
 3 "允许加载文件类型插件
 4 filetype plugin on
 5 "不同的文件定义不同的缩进格式
 6 filetype indent on
 7
 8 "设置颜色，这里有多种颜色可以配置
 9 let colors_name = "darkblue"
10
11 syntax enable
12 syntax on
13
14 "打开行标
15 set nu
16
17 "tab 长度 4 个空格
18 set tabstop=4
19
20 set softtabstop=4
21 set shiftwidth=4
22 set noexpandtab
23
24 "设置编码自动识别
25 set fileencodings=utf-8,gbk
26 set ambiwidth=double
27
28 "启动鼠标
29 "set mouse=a
30
31 "不同时显示多个文件的 tag，只显示当前文件的
32 let Tlist_Show_One_File=1
33 "如果 taglist 窗口是最后一个窗口，则退出 vim
34 let Tlist_Exit_OnlyWindow=1
35 "在启动 vim 后，自动打开 taglist 窗口
36 let Tlist_Auto_Open=1
37
38 "一键运行 python
39 map <F5> :!python %
```

（3）保存文件后，来看看效果如何。如图 3-4 所示，执行命令：

```
vi  ~/.vimrc
```

图 3-4　vim IDE

如果编辑的是一个 python 脚本，直接按 F5 键就会执行该脚本。当然，也可以将这种方法扩展到 C、C++、bash……上。无穷的思路，无限的可能。只有想不到，没有做不到。这就是 vim。



注　意

花点时间再花点心思完全可以配置一个为自己量身定做的 IDE。它也许不是最好的，但它一定是最合适的。

3.2.4　vim 使用指南

如果要详细地解说 vim 恐怕几本书都说不完，在这里只是简单解说一下 vim 的基本常用功能。vi 有 3 个模式：插入模式、命令模式、低行模式。

- 插入模式：在此模式下可以输入字符，按 Esc 键将回到命令模式。
- 命令模式：可以移动光标、删除字符等。
- 低行模式：保存文件、退出 vi、设置 vi、查找等功能（低行模式也可以看作是命令模式里的）。

（1）打开文件、保存文件、关闭文件（vi 命令模式下使用）

```
vi filename       //打开 filename 文件
:w                //保存文件
:w vpser.net      //保存至 vpser.net 文件
:q                //退出编辑器，如果文件已修改请使用下面的命令
:q!               //退出编辑器，且不保存
:wq               //退出编辑器，且保存文件
```

（2）插入文本或行（vi 命令模式下使用，执行下面命令后将进入插入模式，按 Esc 键可退出插入模式）

```
a               //在当前光标位置的右边添加文本
i               //在当前光标位置的左边添加文本
A               //在当前行的末尾位置添加文本
I               //在当前行的开始处添加文本（非空字符的行首）
O               //在当前行的上面新建一行
o               //在当前行的下面新建一行
R               //替换（覆盖）当前光标位置及后面的若干文本
J               //合并光标所在行及下一行为一行（依然在命令模式）
```

（3）移动按键（vi 命令模式下使用）

- 使用上下左右方向键
- h 向左、j 向下、k 向上、l 向右。
- 空格键向右、Backspace 键向左、Enter 键移动到下一行首、-键移动到上一行首。

（4）删除、恢复字符或行（vi 命令模式下使用）

```
x               //删除当前字符
nx              //删除从光标开始的 n 个字符
dd              //删除当前行
ndd             //向下删除当前行在内的 n 行
u               //撤销上一步操作
U               //撤销对当前行的所有操作
```

（5）搜索（vi 命令模式下使用）

```
/vpser          //向光标下搜索 vpser 字符串
?vpser          //向光标上搜索 vpser 字符串
n               //向下搜索前一个搜索动作
N               //向上搜索前一个搜索动作
```

（6）跳至指定行（vi 命令模式下使用）

```
n+              //向下跳 n 行
n-              //向上跳 n 行
nG              //跳到行号为 n 的行
G               //跳至文件的底部
```

（7）设置行号（vi 命令模式下使用）

```
:set  nu            //显示行号
:set nonu           //取消显示行号
```

（8）复制、粘贴（vi 命令模式下使用）

```
yy              //将当前行复制到缓存区，也可以用 "ayy 复制，"a 为缓冲区，a 也可以替换为 a~z 的任
意字母，可以完成多个复制任务。
nyy             //将当前行向下 n 行复制到缓冲区，也可以用 "anyy 复制，"a 为缓冲区，a 也可以替换
为 a~z 的任意字母，可以完成多个复制任务。
yw              //复制从光标开始到词尾的字符。
nyw             //复制从光标开始的 n 个单词。
```

```
y^          //复制从光标到行首的内容。
y$          //复制从光标到行尾的内容。
p           //粘贴剪切板里的内容在光标后，如果使用了前面的自定义缓冲区，建议使用"ap 进行粘贴。
P           //粘贴剪切板里的内容在光标前，如果使用了前面的自定义缓冲区，建议使用"aP 进行粘贴。
```

（9）替换（vi 命令模式下使用）

```
:s/old/new          //用 new 替换行中首次出现的 old
:s/old/new/g        //用 new 替换行中所有的 old
:n,m s/old/new/g    //用 new 替换从 n~m 行里所有的 old
:%s/old/new/g       //用 new 替换当前文件里所有的 old
```

（10）编辑其他文件

```
:e otherfilename    //编辑文件名为 otherfilename 的文件。
```

（11）修改文件格式

```
:set fileformat=unix        //将文件修改为 unix 格式，如 win 下面的文本文件在 Linux 下会出
现^M
```

好了，vim 就介绍到这里。vim 是个非常有用的文本编辑器，掌握 vim 是学习 Linux 的重要一环。



注 意

vim 用于局部的搜索、替换、复制、粘贴，非常方便，这是在编写代码时的常用功能。在这方面其他文本编辑器难以比拟。

3.3 bash

使用 Putty 登录 Raspberry。登录 Raspberry 后使用的就是 bash。使用其他的 Linux PC，开机后登录桌面，桌面也是作用于 bash 上的。本章将要介绍的就是 bash。

3.3.1 bash 简介

Shell 是使用者和 Linux 内核的中间层。使用者通过 Shell 与内核（kernel）来沟通，让 kernel 可以控制硬件工作。Windows、Linux、Mac OS 都有 shell。

Shell 的本意是外壳，是包裹内核的壳子的意思。Shell 的版本有很多，常见的有 GNU bash、C shell、K shell。而常说的 Linux bash 大多数都是指 GNU Bash（GNU Bourne-Again Shell），它是 Bourne Shell 的增强版本。

Bash 是一门解释型语言，就是说它的程序不是通过编译后执行的，它是边解释边执行。这种解释型语言的缺点是执行效率比较低，但兼容性比较好。Bash 功能强大，虽然不能与 python、perl、ruby 相比，但尺有所短寸有所长，它胜在简单方便，基本常用问题都可以用 bash 来解决，所以经久不衰。

Bash 是 GNU 计划中重要的工具软件之一，目前也是 Linux distributions 的标准 shell。Bash 的优点主要有以下几点。

1. 命令回溯（history）

用户登录系统后可以查看 ~/.bash_history 文件来查询上次登录时使用过的命令，也可以利用 history [number]命令来查看本次登录使用的命令。例如查看最近执行的 10 个命令，执行命令：

```
history 10
```

2. 命令与文件补全（Tab 键补全）

Tab 键在 Linux 中是个非常方便的按键。如果有个命令，只记得是以 ar 开头，不记得全名是什么。不要着急，按几下 Tab 键，它会把所有以 ar 开头的命令都显示出来，如图 3-5 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ ar
ar                              arm-linux-gnueabihf-gprof
arch                            arm-linux-gnueabihf-ld
arecord                         arm-linux-gnueabihf-ld.bfd
arecordmidi                     arm-linux-gnueabihf-ld.gold
arm-linux-gnueabihf-addr2line   arm-linux-gnueabihf-nm
arm-linux-gnueabihf-ar          arm-linux-gnueabihf-objcopy
arm-linux-gnueabihf-as          arm-linux-gnueabihf-objdump
arm-linux-gnueabihf-c++filt     arm-linux-gnueabihf-ranlib
arm-linux-gnueabihf-cpp         arm-linux-gnueabihf-readelf
arm-linux-gnueabihf-cpp-4.6     arm-linux-gnueabihf-run
arm-linux-gnueabihf-dwp         arm-linux-gnueabihf-size
arm-linux-gnueabihf-elfedit     arm-linux-gnueabihf-strings
arm-linux-gnueabihf-g++         arm-linux-gnueabihf-strip
arm-linux-gnueabihf-g++-4.6     arp
arm-linux-gnueabihf-gcc         arpd
arm-linux-gnueabihf-gcc-4.6
pi@raspberrypi ~ $ ar
```

图 3-5　bash 命令补全演示 1

如果想用 vim 编辑当前目录下以 g 开头的文件，也很简单，执行命令 vi Tree。然后再按几下 Tab 键，它会列出当前目录下所有以 g 开头的文件，如图 3-6 所示。

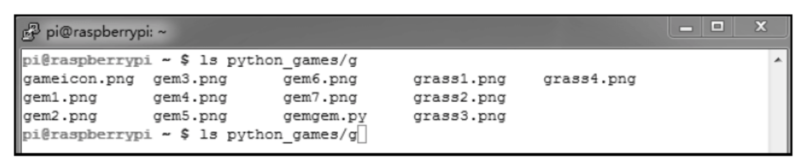


图 3-6　bash 命令补全演示 2

3. 命令别名（alias）

想用 vi 命令来调用 vim，除了把 vi 命令链接到 vim 上还有没有其他的方法呢？当然有，这里可以使用命令别名。执行命令：

```
Alias'vi=vim'
```

有点类似于给命令起个“绰号”，实际上还是同一个命令。但是显然“绰号”更容易记忆。在 Linux 中 alias 不光是起个“绰号”的作用，它还可以用于命令合成。例如：我们通常使用 ls 命令查看文件。但如果想详细查看某个文件，就得使用“ls–l”命令了。这里我们可以合成一个命令 ll。展示效果如图 3-7 所示，执行命令：

```
alias ll='ls -l'
```




```
pi@raspberrypi ~ $ touch abc.txt
pi@raspberrypi ~ $ ls abc.txt
abc.txt
pi@raspberrypi ~ $ ll abc.txt
-bash: ll: command not found
pi@raspberrypi ~ $ alias ll='ls -l'
pi@raspberrypi ~ $ ls -l abc.txt
-rw-r--r-- 1 pi pi 0 Sep 24 22:09 abc.txt
pi@raspberrypi ~ $ ll abc.txt
-rw-r--r-- 1 pi pi 0 Sep 24 22:09 abc.txt
pi@raspberrypi ~ $
```

图 3-7　alias 演示 1

如果不喜欢 E 文的日期显示方式，也可以利用 alias 将合成的 ll 命令改成显示中文日期，再来测试一下，如图 3-8 所示，执行命令：

```
alias ll='ls --color=auto -l --time-style="+%Y/%m/%d %H:%M:%S"'
```

```
pi@raspberrypi ~ $ alias ll='ls --color=auto -l --time-style="+%Y/%m/%d %H:%M:%S
"'
pi@raspberrypi ~ $ ls abc.txt
abc.txt
pi@raspberrypi ~ $ ls -l abc.txt
-rw-r--r-- 1 pi pi 0 Sep 24 22:09 abc.txt
pi@raspberrypi ~ $ ll abc.txt
-rw-r--r-- 1 pi pi 0 2015/09/24 22:09:18 abc.txt
pi@raspberrypi ~ $
```

图 3-8　alias 演示 2

4. 工作前台后台控制（jobs、fg、bg）

在 bash 中执行的任务可以分为前台、后台的。前台的任务是可见的，后台的任务是隐藏的。使用 jobs 命令，可以看到后台的任务及编号。然后使用 fg number 命令，将任务调到前台来执行。也可以将当前执行的前台任务按 Ctrl + Z 组合键后暂停，再使用 bg number 命令，将任务调到后台执行。前台任务演示，如图 3-9 所示。

```
pi@raspberrypi ~ $ wget --limit-rate=300k http://cdimage.debian.org/debian-cd/8.
1.0/amd64/iso-dvd/debian-8.1.0-amd64-DVD-1.iso
--2015-08-28 16:33:51--  http://cdimage.debian.org/debian-cd/8.1.0/amd64/iso-dvd
/debian-8.1.0-amd64-DVD-1.iso
Resolving cdimage.debian.org (cdimage.debian.org)... 130.239.18.173, 130.239.18.
165, 2001:6b0:e:2018::173, ...
Connecting to cdimage.debian.org (cdimage.debian.org)|130.239.18.173|:80... conn
ected.
HTTP request sent, awaiting response... 302 Found
Location: http://saimei.acc.umu.se/debian-cd/8.1.0/amd64/iso-dvd/debian-8.1.0-am
d64-DVD-1.iso [following]
--2015-08-28 16:33:58--  http://saimei.acc.umu.se/debian-cd/8.1.0/amd64/iso-dvd/
debian-8.1.0-amd64-DVD-1.iso
Resolving saimei.acc.umu.se (saimei.acc.umu.se)... 130.239.18.138, 2001:6b0:e:20
18::138
Connecting to saimei.acc.umu.se (saimei.acc.umu.se)|130.239.18.138|:80... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 3994091520 (3.7G) [application/x-iso9660-image]
Saving to: `debian-8.1.0-amd64-DVD-1.iso'

 0% [                                   ] 118,300     25.3K/s  eta 42h 47m
```

图 3-9　前台任务

按 Ctrl + Z 组合键将未完成的前台任务暂停，如图 3-10 所示。

```
 0% [                                       ] 250,404     23.6K/s  eta 2d 0h   ^
Z
[1]+  Stopped                 wget --limit-rate=300k http://cdimage.debian.org/d
ebian-cd/8.1.0/amd64/iso-dvd/debian-8.1.0-amd64-DVD-1.iso
pi@raspberrypi ~ $
```

图 3-10　暂停任务

使用 bg 命令将暂停的任务放入后台执行，如图 3-11 所示。

```
pi@raspberrypi: ~
Connecting to cdimage.debian.org (cdimage.debian.org)|130.239.18.173|:80... conn
ected.
HTTP request sent, awaiting response... 302 Found
Location: http://saimei.acc.umu.se/debian-cd/8.1.0/amd64/iso-dvd/debian-8.1.0-am
d64-DVD-1.iso [following]
--2015-08-28 16:33:58--  http://saimei.acc.umu.se/debian-cd/8.1.0/amd64/iso-dvd/
debian-8.1.0-amd64-DVD-1.iso
Resolving saimei.acc.umu.se (saimei.acc.umu.se)... 130.239.18.138, 2001:6b0:e:20
18::138
Connecting to saimei.acc.umu.se (saimei.acc.umu.se)|130.239.18.138|:80... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 3994091520 (3.7G) [application/x-iso9660-image]
Saving to: `debian-8.1.0-amd64-DVD-1.iso'

 0% [                                       ] 250,404     23.6K/s  eta 2d 0h   ^
Z
[1]+  Stopped                 wget --limit-rate=300k http://cdimage.debian.org/d
ebian-cd/8.1.0/amd64/iso-dvd/debian-8.1.0-amd64-DVD-1.iso
pi@raspberrypi ~ $ bg 1
[1]+ wget --limit-rate=300k http://cdimage.debian.org/debian-cd/8.1.0/amd64/iso-
dvd/debian-8.1.0-amd64-DVD-1.iso &
 0% [                                       ] 268,032      706B/s  eta 55d 23h p
 0% [                                       ] 484,956     --.-K/s  eta 32d 15h
```

图 3-11　后台任务

使用 fg 命令将暂停的任务放到前台来执行，如图 3-12 所示。

```
 0% [                                       ] 484,956     --.-K/s  eta 51d 18h ^
Z
[1]+  Stopped                 wget --limit-rate=300k http://cdimage.debian.org/d
ebian-cd/8.1.0/amd64/iso-dvd/debian-8.1.0-amd64-DVD-1.iso
pi@raspberrypi ~ $ fg 1
wget --limit-rate=300k http://cdimage.debian.org/debian-cd/8.1.0/amd64/iso-dvd/d
ebian-8.1.0-amd64-DVD-1.iso
 0% [                                       ] 484,956     --.-K/s  eta 52d 15h
```

图 3-12　后台任务调入前台

5. 通配符（wildcard）&特殊符号

在 bash 的操作环境中还有一个非常有用的功能，那就是通配符（wildcard）。我们利用 bash 处理数据就更方便了。下面列出一些常用的通配符：

- *：代表“0 个到无穷多个”任意字符。
- ?：代表“一定有一个”任意字符。
- []：同样代表一定有一个在括号内的字符（非任意字符）。例如[abcd]代表一定有一个字符，可能是 a、b、c、d 这 4 个中的任何一个。
- [-]：若有减号在中括号内，代表“在编码顺序内的所有字符”。例如[0-9]代表 0~9 的所

有数字，因为数字的语系编码是连续的。

- ^：若中括号内的第一个字符为指数符号（^），表示“反向选择”，例如[^abc]代表一定有一个字符，只要是非 a、b、c 的其他字符就接受的意思。

以下列出 bash 环境中的特殊符号：

- #：注释符号，最常使用在 script 中，视为说明，在后的数据均不运行。
- \：转义符号，将特殊字符或通配符还原成一般字符。
- |：管道（pipe），分隔两个管道命令的界定。
- ;：连续命令分隔符，连续性命令的界定（注意，与管道命令并不相同）。
- ~：用户的家目录。
- $：取用变量前导符，变量之前需要加的变量取代值。
- &：工作控制（job control），将命令变成后台下工作。
- !：逻辑运算中的逻辑非，not 的意思。
- /：目录符号，路径分隔的符号'。
- >,>>：数据流重定向，输出重定向。
- <,<<：数据流重定向，输入重定向。
- '：单引号，不具有变量置换的功能。
- "：具有变量置换的功能。
- ``：两个“`”中间为可以先运行的命令。
- ()：在中间为子 shell 的起始与结束。
- {}：在中间为命令区块的组合。

创建文件时，尽量不要使用以上的字符做文件名。

6. bash script

有时完成一些简单的任务，使用 python、perl 未免有杀鸡用牛刀的感觉，但一些重复性的工作，一条命令一条命令地敲实在是很痛苦的事情，所以还是用 bash script 吧。bash script 实际就是将一条条的 bash 命令组合起来，然后再一起执行。

3.3.2 第一个 bash 脚本 Hello world

几乎所有的编程语言都是以 hello world 开始的，尊重传统。下面就开始第一个 bash 脚本 hello.sh。

1. 创建工作平台

通过 Putty 登录 Raspberry 后，首先创建一个工作目录，执行命令：

```
mkdir -pv ~/code/bash
cd ~/code/bash
```

第一条命令是在用户家目录下建立工作目录~/code/bash，第二条命令是进入该目录。因为是以

默认用户 pi 登录。所以，建立的目录位置是/home/pi/code/bash/

2. 使用 vim 创建 hello.sh

Linux 不像 Windows 那样需要靠后缀名来确定打开文件的程序，所以 bash script 的后缀名叫什么都可以，只要符合文件名的规则就行。但一般都是把 bash script 的后缀名设定为 sh。以便确认该文件的编译语言。执行命令：

```
vi hello.sh
```

以下是 hello.sh 的代码：

```
1 #!/bin/bash
2
3 echo "Hello world!"
4 printf "This is my first bash script!\n"
```



注 意

前面的 1、2、3、4 是 vim 自动标出的行号，不是代码内容，如果不需要行号，可以按 Esc 键后，输入:set nonu 来取消行号。

执行命令 sh hello.sh，如图 3-13 所示。

```
pi@raspberrypi: ~/code/bash
pi@raspberrypi ~/code/bash $ sh hello.sh
Hello world!
This is my first bash script!
pi@raspberrypi ~/code/bash $
```

图 3-13　执行脚本

这里要说明的是第一行，#!/bin/bash 是指明解释器的位置。其中，#!是一个特殊的表示符，其后，跟着解释此脚本的 shell 路径。也就是说，如果想用 ksh、tsh 来解释脚本。那么第一行就应该是#!/bin/ksh 或者#!/bin/tsh。

后面的 echo 和 printf 都是打印输出命令。篇幅有限，如果对具体语法有兴趣，请自行参考谷歌百度。

再来看一下 sh 这个命令，在 Ubuntu 和 Debian 中 sh 默认指向的不是/bin/bash，而是/bin/dash。Dash 比 bash 速度更快，语法严格遵守 POSIX 标准，但功能比 bash 少很多。一般读者学习的都是 bash，写的 script 脚本也是以 bash 写的。为了防止一些莫名其妙的错误，解决方法有 2 个。

- 执行命令的时候直接用 bash 命令。

比如上例中的 hello.sh 文件，如果将它赋予执行权限，直接运行命令./hello.sh 来执行是没问题的。如果非要带上执行的命令，那就运行命令 bash hello.sh。

- 将 sh 的链接指向 bash。

把 sh 指向 bash，一劳永逸地解决问题。推荐使用这个方法。至于 dash 简洁方便的优点，以现在计算机的性能来说，实在是不算什么。SO，运行命令：

```
sudo rm /bin/sh
sudo ln -s /bin/bash /bin/sh
```



注 意

在一般情况下 dash 和 bash 是可以互换使用的，写 srcipt 时则不行。例如 for 语句，dash 和 bash 就完全不一样。所以没什么特别缘由还是换成 bash 吧。

3.3.3 bash script 实例——增量备份脚本

在上文 2.4.2 中学习过增量备份系统。但每次都输入命令太麻烦了，而且这还是个重复性的工作，可以交给 shell 去完成。下面就写一个增量备份的 bash 脚本 backSYS.sh。代码如下：

```
 1 #!/bin/bash
 2
 3 ### set user variable <<<
 4 ### 备份目录，这里设置的是/mnt/disk 目录下备份，要先将一块磁盘挂载到/mnt/disk 下
 5 backDir="/mnt/disk/backup/raspberry/"
 6 ### 备份文件名
 7 firstName="origin"
 8 ### 后缀名
 9 endName=".tar.gz"
10 ### 增量名
11 secName="_incremental_"
12 ### 不备份的文件夹
13 notBack=" --exclude=/proc --exclude=/lost+found --exclude=/mnt --exclude=/sys
--exclude=/tmp"
14 ### user variable over >>>
15
16 function createOrigin()
17 {
18     rm ${firstName}${secName}*${endName}
19     rm snapshot
20     echo -e "开始创建初始基准备份文件 \t"$firstName$endName
21     time tar -g snapshot -zcpvf `echo ${firstName}${endName}` / --exclude=/proc
--exclude=/lost+found --exclude=/mnt --exclude=/sys --exclude=/tmp
22     echo $?
23     echo -e "初始基准备份文件创建完毕 \t"$firstName$endName
24 }
25
26 function createInc()
27 {
28     echo "开始创建增量备份文件"
29     for i in {1..100};
30     do
31         incName=${firstName}${secName}"${i}"${endName};
```

```
32          if [ -f ${incName} ];
33          then
34              echo "${incName} 已存在"
35              continue
36          else
37              echo "创建增量备份文件 ${incName}"
38               time tar -g snapshot -zcpvf `echo ${incName}` /  --exclude=/proc
--exclude=/lost+found --exclude=/mnt --exclude=/sys --exclude=/tmp
39              echo $?
40              echo -e "增量备份文件创建完毕\t"${incName}
41              break
42          fi
43      done
44 }
45
46 if [ -d ${backDir} ];
47 then
48     echo -e ${backDir}"\t 备份目录已创建"
49 else
50     mkdir -pv ${backDir}
51 fi
52
53 cd ${backDir}
54
55 if [ -f ${firstName}${endName} ];
56 then
57     echo -e ${firstName}${endName}"\t 初始基准备份文件已创建"
58     ls -al ${firstName}${endName}
59     createInc
60 else
61     createOrigin
62 fi
```

好了，以后如果想备份系统，就直接运行命令：

```
sudo sh backSYS.sh
```

它将自动地增量备份系统。如果想添加其他的功能，可以自行添加。经过不断地修正改进，说不定它会成为下一个 Ghost for Linux。



注 意

系统的备份是非常有必要的。强烈建议至少在系统安装完成后备份一次，配置完成后再备份一次。

3.4 Python

Python 和 Perl 是目前最流行的脚本语言，它们各有拥趸。比较而言，python 更加简单。目前 python 在国内更加流行，使用者也更多，本章将介绍 python 语言。

3.4.1 Python 简介

Python 是种解释型语言。也就是说 python script 是无须编译的。与 bash 不同的是它功能强大，借助于丰富的标准库和第三方模块，它在功能上毫不逊色于 C、C++、Java……老牌编程语言。

Python 由 Guido van Rossum 于 1989 年底发明，第一个公开发行版发行于 1991 年。Python 源代码同样遵循 GPL（GNU General Public License）协议。Python 语法简洁而清晰，具有丰富和强大的类库。它常被昵称为胶水语言(glue language)，能够把用其他语言制作的各种模块(尤其是 C/C++)很轻松地联结在一起。常见的一种应用情形是，使用 Python 快速生成程序的原型（有时甚至是程序的最终界面），然后对其中有特别要求的部分，用更合适的语言改写。

目前 Python 的主流版本是 Python 2.7 和 Python 3.4。个人感觉 Python 2.7 编程风格偏向于 C 面向程序编程。而 Python 3.4 则偏向于 C++面向对象编程。当然，也可以把 Python 2.7 写成 C++风格，把 Python 3.4 写成 C 风格，只要语法没错误，随便怎么写都可以。Python 3.4 是目前最新的版本，但 Python 3.4 的中文参考资料还不多。使用者比较少，一般都是使用的 Python 2.7。下文中，如果没有特殊说明，所指的 Python 都是 Python 2.7。

Python 开发总的指导思想是，对于一个特定的问题，只要有一种最好的方法来解决就好了。这在由 Tim Peters 写的 Python 格言（称为 The Zen of Python）里面表述为：There should be one-- and preferably only one --obvious way to do it。 这正好和 Perl 语言（另一种功能类似的高级动态语言）的中心思想 TMTOWTDI（There's More Than One Way To Do It）完全相反。

Python 的作者有意地设计限制性很强的语法，使得不好的编程习惯（例如 if 语句的下一行不向右缩进）都不能通过编译。其中很重要的一项就是 Python 的缩进规则。一个和其他大多数语言（如 C）的区别就是，一个模块的界限，完全是由每行的首字符在这一行的位置来决定的（而 C 语言是用一对花括号{}来明确地定出模块的边界的，与字符的位置毫无关系）。不可否认的是，通过强制程序员们缩进（包括 if、for 和函数定义等所有需要使用模块的地方），Python 确实使得程序更加清晰和美观。

Python 本身被设计为可扩充的。并非所有的特性和功能都集成到语言核心。Python 提供了丰富的 API 和工具，以便程序员能够轻松地使用 C 语言、C++、Cython 来编写扩充模块。Python 编译器本身也可以被集成到其他需要脚本语言的程序内。因此，很多人还把 Python 作为一种“胶水语言”使用。使用 Python 将其他语言编写的程序进行集成和封装。在 Google 内部的很多项目，例如 Google Engine 使用 C++编写性能要求极高的部分，然后用 Python 或 Java/Go 调用相应的模块。

3.4.2 第一个 Python 脚本 Hello world

还是尊重传统，写第一个 Python script，hello.py。Python script 的后缀名一般都是.py。至于原因，前文已经说明不再赘述。

1. 创建工作目录

通过 Putty 登录 Raspberry 后，首先创建一个工作目录，执行命令：

```
mkdir -pv ~/code/python/hello
```

```
cd ~/code/python/hello
```

2. 使用 vim 创建 hello.py

使用 vim 编辑 hello.py，执行命令：

```
vi hello.py
```

以下是 hello.py 的代码：

```
 1 #!/usr/bin/env python
 2 # -*- coding:utf-8 -*-
 3 #Author  :hstking
 4 #E-mail  :hstking@hotmail.com
 5 #Ctime   :2015/08/05
 6 #Mtime   :
 7 #Version :
 8
 9
10 def main():
11     print "Hello world\n"
12
13
14 if __name__ == '__main__':
15    main()
```

只有 15 行，看起来很简单是不是？还是来解释一下。每行前面的数字行标就不解释了。第一行的#!特殊标示符前文 3.3.2 中也解释过了。还记得 bash script 吗？#!后面跟的是解释器的位置，这里也是一样的。只是 bash 的位置总是在/bin/bash 里，而 Python 却不一样。而且，有时需要用 Python 3 来解释脚本，有时需要用 Python 2.7 来解释脚本。所以直接使用/usr/bin/env 命令，让它自己在环境变量中去寻找用哪个 Python 来解释。至于这个 Python 是指的哪个脚本呢？如图 3-14 所示。

```
pi@raspberrypi: ~/code/bash
pi@raspberrypi ~/code/bash $ whereis python
python: /usr/bin/python3.2mu /usr/bin/python3.2 /usr/bin/python /usr/bin/python2
.7 /usr/bin/python2.6 /usr/bin/python2.7-config /etc/python3.2 /etc/python /etc/
python2.7 /etc/python2.6 /usr/lib/python3.2 /usr/lib/python2.7 /usr/lib/python2.
6 /usr/bin/X11/python3.2mu /usr/bin/X11/python3.2 /usr/bin/X11/python /usr/bin/X
11/python2.7 /usr/bin/X11/python2.6 /usr/bin/X11/python2.7-config /usr/local/lib
/python3.2 /usr/local/lib/python2.7 /usr/local/lib/python2.6 /usr/include/python
3.2mu /usr/include/python2.6_d /usr/include/python2.7 /usr/include/python2.6 /us
r/include/python3.2dmu /usr/include/python2.7_d /usr/share/python /usr/share/man
/man1/python.1.gz
pi@raspberrypi ~/code/bash $ ls -l /usr/bin/python
lrwxrwxrwx 1 root root 9 Sep 29  2013 /usr/bin/python -> python2.7
pi@raspberrypi ~/code/bash $
```

图 3-14　Python 位置

脚本中第 2 行指定了字符编码。如果没什么特别要求，一般都是指定 utf-8。第 3 行是作者信息。第 4 行是 mail。第 5 行是创建时间。第 6 行是最后修改时间。第 7 行是版本号。第 10 行定义 main 函数。第 11 行是 main 函数体。第 14 行是在测试该文件时，当成脚本执行还是当成第三方模块调用。第 15 行调用 main 函数。

只是打印一个简单的 Hello world!居然用了 15 行。需要这么复杂吗？当然不用，实际上只需要 3 行就可以达到同样的效果。其他多余的行都是注释，是通过一个简单的 python 脚本 touch2py.py 自动创建的。下文再来详细讲解 touch2py.py。这里还是先看下最简版本的 Hello world 脚本 h0.py。以下是 h0.py 的代码：

```
1 #!/usr/bin/env python
2
3 print "Hello world!\n"
```

加上空行，只需要 3 行就够了。

3. 执行结果

比较这两个脚本的执行结果，结果如图 3-15 所示，执行命令：

```
python hello.py
python h0.py
```

```
pi@raspberrypi: ~/code/python/hello
pi@raspberrypi ~/code/python/hello $ ls -l
total 8
-rw-r--r-- 1 pi pi  45 Aug 29 12:43 h0.py
-rw-r--r-- 1 pi pi 210 Aug  7 21:02 hello.py
pi@raspberrypi ~/code/python/hello $ python hello.py
Hello world

pi@raspberrypi ~/code/python/hello $ python h0.py
Hello world!

pi@raspberrypi ~/code/python/hello $
```

图 3-15 hello.py&h0.py

结果完全一样。至于有没有必要加那么多的注释。就是仁者见仁，智者见智了。我的看法是，“不管你觉得需不需要，反正我需要”。



注 意

写代码添加注释，这是一个良好的习惯。如果不想以后绞尽脑汁的回忆这个函数起什么作用，那个变量代表什么。还是留下注释吧。

3.4.3 Python 常用模块

Python 的标准库不多，但加上第三方模块那就多得可怕了。同一种功能可能有多个不同的模块，足够满足生产中的各种问题。如果觉得还不够，也可以自己编写模块加入其中。一般来说下面的模块已经足够使用了。没必要重复地造轮子。

1. Python 运行时服务

- copy：copy 模块提供了对复合（compound）对象（list，tuple，dict，custom class）进行浅拷贝和深拷贝的功能。
- pickle：pickle 模块被用来序列化 python 的对象到 bytes 流，从而适合存储到文件、网络传输或数据库存储。（pickle 的过程也称为 serializing、marshalling 或者 flattening，pickle

同时可以用来将 bytes 流反序列化为 python 的对象）。

- sys：sys 模块包含了与 python 解析器和环境相关的变量和函数。
- 其他：atexit、gc、inspect、marshal、traceback、types、warnings、weakref。

2. 数学

- decimal：python 中的 float 是使用双精度的二进制浮点编码来表示的，这种编码导致了小数不能被精确地表示，例如 0.1 实际上内存中为 0.100000000000000001，还有 3*0.1 == 0.3 为 False。decimal 就是为了解决类似的问题的，拥有更高的精确度，能表示更大范围的数字，更精确地四舍五入。
- math：math 模块定义标准的数学方法，例如 cos（x）、sin（x）等。
- random：random 模块提供各种方法用来产生随机数。
- 其他：fractions，numbers。

3. 数据结构、算法和代码简化

- array：array 代表数组，类似于 list，与 list 不同的是只能存储相同类型的对象。
- bisect：bisect 是一个有序的 list，其中内部使用二分法（bitsection）来实现大部分操作。
- collections：collections 模块包含了一些有用的容器的高性能实现，各种容器的抽象基类和创建 name-tuple 对象的函数。例如包含了容器 deque、defaultdict、namedtuple 等。
- heapq：heapq 是一个使用 heap 实现的带有优先级的 queue。
- itertools：itertools 包含的函数用来创建有效的 iterators。所有的函数都返回 iterators，或者函数包含 iterators（例如 generators 和 generators expression）。
- operator： operator 提供了访问 python 内置的操作和解析器提供的特殊方法，例如 x+y 为 add（x，y），x+=y 为 iadd（x，y），a % b 为 mod（a，b）等等。
- 其他：abc、contextlib、functools。

4. string 和 text 处理

- codecs：codecs 模块用来处理不同的字符编码与 unicode text io 的转化。
- re：re 模块用来对字符串进行正则表达式的匹配和替换。
- string：string 模块包含大量有用的常量和函数用来处理字符串，也包含新字符串格式的类。
- struct：struct 模块用来在 python 和二进制结构间实现转化。
- unicodedata：unicodedata 模块提供访问 unicode 字符数据库。

5. Python 数据库访问

关系型数据库拥有共同的规范 Python Database API Specification V2.0，MySQL、Oracle 等都实现了此规范，然后增加自己的扩展。

- sqlite3：sqlite3 模块提供 SQLite 数据库访问的接口。SQLite 数据库是以一个文件或内存的形式存在的自包含的关系型数据库。

- DBM-style 数据库模块：Python 提供大量的 modules 来支持 UNIX DBM-style 数据库文件。dbm 模块用来读取标准的 UNIX-dbm 数据库文件，gdbm 用来读取 GNU dbm 数据库文件，dbhash 用来读取 Berkeley DB 数据库文件。所有这些模块提供了一个对象实现基于字符串的持久化的字典，它与字典 dict 非常相似，但是它的 keys 和 values 都必须是字符串。
- shelve：shelve 模块使用特殊的 shelf 对象来支持持久化对象。这个对象的行为与 dict 相似，但是所有的其他存储的对象都使用基于 hashtable 的数据库（dbhash、dbm、gdbm）存储在硬盘。与 dbm 模块的区别是所存储的对象不仅是字符串，而且可以是任意的与 pickle 兼容的对象。

6. 文件和目录处理

- bz2：bz2 模块用来处理以 bzip2 压缩算法压缩的文件。
- filecmp：filecmp 模块提供函数来比较文件和目录。
- fnmatch：fnmatch 模块提供使用 UNIX shell-style 的通配符来匹配文件名。这个模块只是用来匹配，使用 glob 可以获得匹配的文件列表。
- glob：glob 模块返回某个目录下与指定的 UNIX shell 通配符匹配的所有文件。
- gzip：gzip 模块提供类 GzipFile，用来执行与 GNUgzip 程序兼容的文件的读写。
- shutil：shutil 模块用来执行更高级别的文件操作，例如复制、删除、改名。shutil 操作针对一般的文件，不支持 pipes、block devices 等文件类型。
- tarfile：tarfile 模块用来维护 tar 存档文件。tar 没有压缩的功能。
- tempfile：tempfile 模块用来产生临时文件和文件名。
- zipfile：zipfile 模块用来处理 zip 格式的文件。
- zlib：zlib 模块提供对 zlib 库的压缩功能的访问。

7. 操作系统的服务

- cmmands：commands 模块被用来执行简单的系统命令，命令以字符串的形式传入，且同时以字符串的形式返回命令的输出。但是此模块只在 UNIX 系统上可用。
- configParser：configParser 模块用来读写 Windows 的 ini 格式的配置文件。
- datetime：datetime 模块提供了各种类型来表示和处理日期和时间。
- errno：定义所有的 errorcode 对应的符号名字。
- io：io 模块实现各种 IO 形式和内置的 open()函数。
- logging: logging 模块灵活方便地对应用程序记录 events、errors、warnings 和 debuging 信息。这些 log 信息可以被收集、过滤，写到文件或系统 log，甚至通过网络发送到远程的机器上。
- mmap: mmap 模块提供内存映射文件对象的支持，使用内存映射文件与使用一般的文件或 byte 字符串相似。
- msvcrt：mscrt 只可以在 Windows 系统使用，用来访问 Visual C 运行时库的很多有用的功能。

- optparse：optparse 模块提供更高级别来处理 UNIX style 的命令行选项 sys.argv。
- os：os 模块对通用的操作系统服务提供可移植（portable）的接口。os 可以认为是 nt 和 posix 的抽象。nt 提供 Windows 的服务接口，posix 提供 UNIX（Linux，mac）的服务接口。
- os.path：os.path 模块以可移植的方式来处理路径相关的操作。
- signal：signal 模块用来实现信号（signal）处理，往往跟同步有关。
- subprocess：subprocess 模块包含函数和对象来统一创建新进程，控制新进程的输入输出流，处理进程的返回。
- time：time 模块提供各种时间相关的函数。常用的是 time.sleep().
- winreg：winreg 模块用来操作 Windows 注册表。
- 其他：fcntl。

8. 线程和并行

- multiprocessing：multiprocessing 模块提供通过 subprocess 来加载多个任务，通信、共享数据，执行各种同步操作。
- threading：threading 模块提供了 thread 类的很多同步方法来实现多线程编程。
- queue：queue 模块实现各种多生产者、多消费者队列，用来实现多线程程序的信息安全交换。
- 其他：Coroutines and Microthreading。

9. 网络编程和套接字（sockets）

- asynchat：asynchat 模块通过封装 asyncore 来简化应用程序的网络异步处理。
- ssl：ssl 模块用来使用 secure sockets layer（SSL）包装 socket 对象，从而实现数据加密和终端认证。python 使用 openssl 来实现此模块。
- socketserver：socketserver 模块提供类型简化了 TCP、UDP 和 UNIX 领域的 socket server 的实现。
- 其他：asyncore，select。

10. internet 应用程序编程

- ftplib：ftplib 模块实现 ftp 的 client 端协议。此模块很少使用，因为 urllib 提供了更高级的接口。
- httplib：httplib 包含 http client 和 server 的实现和 cookies 管理的模块。
- smtplib：smtplib 包含 smtp client 的底层接口，用来使用 smtp 协议发送邮件。
- urllib：urllib 包提供高级的接口来实现与 http server、ftp server 和本地文件交互的 client。
- xmlrpc：xmlrpc 模块被用类实现 XML-RPC client。

11. Web 编程

- cgi：cgi 模块用来实现 cgi 脚本，cgi 程序一般被 webserver 执行，用来处理用户在 form

中的输入，或生成一些动态的内容。当与 cgi 脚本有关的 request 被提交，webserver 将 cgi 作为子进程执行，cgi 程序通过 sys.stdin 或环境变量来获得输入，通过 sys.stdout 来输出。

- webbrowser：webbrowser 模块提供平台独立的工具函数来使用 web browser 打开文档。
- 其他：wsgiref/WSGI （Python Web Server Gateway Interface）。

12. internet 数据处理和编码

- base64：base64 模块提供 base64、base32、base16 编码方式，用来实现二进制与文本间的编码和解码。base64 通常用来对编码二进制数据，从而嵌入到邮件或 http 协议中。
- binascii：binascii 模块提供低级的接口来实现二进制和各种 ASCII 编码的转化。
- csv：csv 模块用来读写 comma-separated values（CSV）文件。
- email：email 包提供大量的函数和对象来使用 MIME 标准表示、解析和维护 email 消息。
- hashlib：hashlib 模块实现各种 secure hash 和 message digest algorithms，例如 MD5 和 SHA1。
- json：json 模块用于类序列化或反序列化 Javascript object notation（JSON）对象。
- xml：xml 包提供各种处理 xml 的方法。

这么多的模块，一般来说是足够满足需要的了。如果不够，百度一下吧，还有很多第三方模块没有列入其中。如果还不够，那就只有自行建立合适的模块了。浏览器打开 https://docs.python.org/2.7/py-modindex.html，这是 Python 2.7 的官方标准库的列表，显示了所有官方模块的详细信息。


注 意

如果不是要满足特别奇怪的要求就不需要自行建轮子了。网络上有大把的先行者写的优秀模块，直接拿来借用就可以了。只要善用谷歌百度就行。

3.4.4 Python script 实例——touch2py.py

创建一个 Python 脚本，有时要加入一些脚本信息。这些脚本信息基本都是一样的。每次重复性输入是件很痛苦的事情。幸好 Python 可以解决这个问题。上文已经简单介绍了一些 Python 模块。现在就通过这些模块来创建一个简单的 Python 脚本 touch2py.py。执行命令：

```
mkdir -pv ~/code/python/touch2py
cd ~/code/python/touch2py
vi touch2py.py
```

以下是 tuoch2py.py 的代码：

```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3
4 import sys
5 import os
```

```
 6 import time
 7
 8 def creatFile(name):
 9     f = open(name,'w')
10     for head in pyHead:
11         f.write(head)
12         f.write('\n')
13     f.write('\n')
14     f.close()
15
16
17
18 if __name__ == '__main__':
19     global pyHead
20     pyHead = [
21 '#!/usr/bin/env python',
22 '# -*- coding:utf-8 -*-',
23 '#Author  :hstking',
24 '#E-mail  :hstking@hotmail.com',
25 '#Ctime   :' + time.strftime("%Y/%m/%d"),
26 '#Mtime   :',
27 '#Version :',
28 '\n\n\n\n',
29 "if __name__ == '__main__':"
30 ]
31
32     if len(sys.argv) == 1:
33         print '请输入文件名\n'
34
35     fileNames = sys.argv[1:]
36     for name in fileNames:
37         if (os.path.isfile(name) or os.path.isdir(name) or os.path.islink
(name)):
38             print name,"已经存在"
39         else:
40             creatFile(name)
```

最后讲一下第4、5、6行的导入模块。Python导入模块有两种方式。比如该脚本中的time.strftime函数的导入。如下所示：

- import time：这种导入是将整个time模块一起导入到脚本中，导入后想使用time.strftime函数，就得带入模块名。就如touch2py.py中的使用方法time.strftime()。
- from time import strftime：这种导入只导入了模块中的某个函数。导入后想使用time.strftime函数，就直接用函数名就可以了，如strftime()。

实际上还有一种导入模块的方法：内建函数__import__()，只是这种写法比较少。就不多做说明了。

好了，touch2py.py已经完成了。再来完善一下，把它写进PATH，以后就可以直接将touch2py.py

当命令执行了。执行命令：

```
sudo ln -s /home/pi/code/python/touch2py/touch2py.py /usr/local/bin/touch2py
```

将 touch2py.py 做了一个链接。以后直接使用命令 touch2py 就可以调用 touch2py.py 脚本了。还原刚才的 hello.py，这个脚本是怎么建立的呢？执行命令：

```
touch2py hello.py
vi hello.py
```

此时 hello.py 的代码如下：

```
 1 #!/usr/bin/env python
 2 # -*- coding:utf-8 -*-
 3 #Author  :hstking
 4 #E-mail  :hstking@hotmail.com
 5 #Ctime   :2015/08/05
 6 #Mtime   :
 7 #Version :
 8
 9
10
11
12 if __name__ == '__main__':
```

好了，相应的位置再插入 3 行，就和 3.4.2 中的 hello.py 一模一样了。这样是不是简单了很多呢？在实际生产中，只要是重复性的工作，都可以想办法让 Python 去完成。它就擅长干这个。一劳永逸，何乐而不为。



注 意

嵌入时间、个人信息是为了便于交流。还可以添加更多的信息，自行发挥吧。

3.4.5 Python 进阶实例——getNip.py

本节的目的是通过 Python 取得自己的 Nip。也就是本机的网络 ip 地址。最常用的取 Nip 的方法是什么呢？我都是直接在百度中输入 IP，然后单击“百度一下”，直接看第一个结果就可以了。用 Python 怎么取得这个结果呢？就和刚才的查询过程一样，无非就是用 python 去取，无须自己动手而已。

1. 创建工作台

```
cd
mkdir -pv code/python/getNip
cd $_
touch2py getNip.py
vi getNip.py
```

2. 编写代码

可以开始编写代码了。getNip.py 的代码如下：

```
 1 #!/usr/bin/env python
 2 # -*- coding:utf-8 -*-
 3 #Author  :hstking
 4 #E-mail  :hstking@hotmail.com
 5 #Ctime   :2015/08/27
 6 #Mtime   :
 7 #Version :
 8
 9 import urllib2
10 import re
11 import os
12
13
14 ####  定义 GetNip 类
15 class GetNip():
16 ####  定义构造函数，可以用于定义类变量
17     def __init__(self):
18         self.logPath = os.path.expanduser('~') + os.sep + 'log'
19         self.nipFile = self.logPath + os.sep + 'Nip.txt'
20         self.Nip = None
21
22         self.getNip()
23         self.writeNip()
24
25 ####  从网络取得本地的公网 IP
26     def getNip(self):
27         urls = 'http://1111.ip138.com/ic.asp'
28         if urllib2.urlopen(urls).geturl() == urls:
29             rawString = urllib2.urlopen(urls).read()
30             self.Nip = re.search(b'\d+\.\d+\.\d+\.\d+',rawString).group()
31             print("Nip = %s"%self.Nip)
32         else:
33             print("未取得本机 NIP")
34
35 ####  将取得的公网 IP 写入指定文件中
36     def writeNip(self):
37         if os.path.isdir(self.logPath):
38             pass
39         else:
40             os.makedirs(self.logPath)
41         with open(self.nipFile,'w') as FP:
42             FP.write(self.Nip)
43
44
45 ####  以下是脚本的主程序
```

```
46 if __name__ == '__main__':
47     nip = GetNip()
```

3. 运行结果

运行脚本，验证执行结果。结果如图 3-16 所示，执行命令：

```
python getNip.py
```

```
pi@raspberrypi: ~/code/python/getNip
pi@raspberrypi ~ $ cd code/python/getNip/
pi@raspberrypi ~/code/python/getNip $ ls
getNip.py
pi@raspberrypi ~/code/python/getNip $ python getNip.py
Nip = 221.235.83.144
pi@raspberrypi ~/code/python/getNip $
```

图 3-16　getNip.py 演示

基本达到了设计结果。初学者在写 Python script 时，首要是做到满足设计需求，然后再来精简代码，简化过程。



注　意

这个 script 只是满足了最基本的设计需求，还有改进空间。个人觉得对一些小的功能，一个功能就写成一个 script，使用时再一次性调用。非常方便。

3.5　常用工具

Python 的确无所不能，可寸有所长，尺有所短，有些时候还有一些特定的工具更加方便。毕竟 Python 追求的是广而博，而特定的工具追求的是专而精。

3.5.1　正则表达式（RE）

在介绍这些工具之前不得不先说说正则表达式了。不只是下面的几个工具支持正则。在 Linux 中很多地方都与正则息息相关。正则表达式，又称正规表示法、常规表示法（Regular Expression，在代码中常简写为 regex、regexp 或 RE），是计算机科学的一个概念。正则表达式使用单个字符串来描述、匹配一系列符合某个句法规则的字符串。不管是在 Windows 还是 Linux 中都是有正则表达式的。而且两者还比较相近，但没能统一实在是令人遗憾。

正则表达式由一些普通字符和一些元字符（metacharacters）组成。普通字符包括大小写的字母和数字，而元字符则具有特殊的含义。以下列出了所有元字符及描述。

- \: 将下一个字符标记为一个特殊字符或一个原义字符或一个向后引用或一个八进制转义符。例如，“\\n” 匹配\n。“\n” 匹配换行符。序列 “\\” 匹配 “\” 而 “\(” 则匹配 “(”。即相当于多种编程语言中都有的 “转义字符” 的概念。
- ^: 匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^也匹配 “\n” 或 “\r” 之后的位置。

- $: 匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，$也匹配“\n”或“\r”之前的位置。
- *: 匹配前面的子表达式任意次。例如，zo*能匹配“z”、“zo”以及“zoo”。*等价于{0,}。
- +: 匹配前面的子表达式一次或多次（大于等于 1 次）。例如，“zo+”能匹配“zo”以及“zoo”，但不能匹配“z”。+等价于{1,}。
- ?: 匹配前面的子表达式零次或一次。例如，“do（es）?”可以匹配“do”或“does”中的“do”。?等价于{0,1}。
- {n}: n 是一个非负整数，匹配确定的 n 次。例如，“o{2}”不能匹配“Bob”中的“o”，但是能匹配“food”中的两个 o。
- {n,}: n 是一个非负整数，至少匹配 n 次。例如，“o{2,}”不能匹配“Bob”中的“o”，但能匹配“foooood”中的所有 o。“o{1,}”等价于“o+”，“o{0,}”则等价于“o*”。
- {n,m}: m 和 n 均为非负整数，其中 n<=m。最少匹配 n 次且最多匹配 m 次。例如，“o{1,3}”将匹配“fooooood”中的前 3 个 o。“o{0,1}”等价于“o?”。请注意在逗号和两个数之间不能有空格。
- ?: 当该字符紧跟在任何一个其他限制符（*、+、?、{n}、{n,}、{n,m}）后面时，匹配模式是非贪婪的。非贪婪模式尽可能少地匹配所搜索的字符串，而默认的贪婪模式则尽可能多地匹配所搜索的字符串。例如，对于字符串“oooo”，“o+?”将匹配单个“o”，而“o+”将匹配所有“o”。
- .: 匹配除“\r\n”之外的任何单个字符。要匹配包括“\r\n”在内的任何字符，请使用像“[\s\S]”的模式。
- （pattern）: 匹配 pattern 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，在 VBScript 中使用 SubMatches 集合，在 JScript 中则使用$0…$9 属性。要匹配圆括号字符，请使用“\（”或“\）”。
- （?:pattern）: 匹配 pattern 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用“或”字符（|）来组合一个模式的各个部分时很有用。例如“industr（?:y|ies）”就是一个比“industry|industries”更简略的表达式。
- （?=pattern）: 正向肯定预查，在任何匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如，“Windows（?=95|98|NT|2000）”能匹配“Windows2000”中的“Windows”，但不能匹配“Windows3.1”中的“Windows”。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。
- （?!pattern）: 正向否定预查，在任何不匹配 pattern 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如“Windows（?!95|98|NT|2000）”能匹配“Windows3.1”中的“Windows”，但不能匹配“Windows2000”中的“Windows”。
- （?<+pattern）: 反向肯定预查，与正向肯定预查类似，只是方向相反。例如，“（?<=95|98|NT|2000）Windows”能匹配“2000Windows”中的“Windows”，但不能匹配“3.1Windows”中的“Windows”。

- （?<!pattern）：反向否定预查，与正向否定预查类似，只是方向相反。例如“（?<!95|98|NT|2000）Windows”能匹配“3.1Windows”中的“Windows”，但不能匹配“2000Windows”中的“Windows”。
- X | y：匹配 x 或 y。例如，“z|food”能匹配“z”或“food”或“zood”（此处请谨慎）。“（z|f）ood”则匹配“zood”或“food”。
- [xyz]：字符集合。匹配所包含的任意一个字符。例如，[abc]可以匹配“plain”中的“a”。
- [^xyz]：负值字符集合。匹配未包含的任意字符。例如，[^abc]可以匹配“plain”中的“plin”。
- [a-z]：字符范围。匹配指定范围内的任意字符。例如，[a-z]可以匹配“a”到“z”范围内的任意小写字母字符。注意：只有连字符在字符组内部时，并且出现在两个字符之间时，才能表示字符的范围；如果出现在字符组的开头，则只能表示连字符本身。
- [^a-z]：负值字符范围。匹配任何不在指定范围内的任意字符。例如，“[^a-z]”可以匹配任何不在“a”到“z”范围内的任意字符。
- \b：匹配一个单词边界，也就是指单词和空格间的位置（即正则表达式的“匹配”有两种概念，一种是匹配字符，一种是匹配位置，这里的\b 就是匹配位置的）。例如，“er\b”可以匹配“never”中的“er”，但不能匹配“verb”中的“er”。
- \B：匹配非单词边界。“er\B”能匹配“verb”中的“er”，但不能匹配“never”中的“er”。
- \cx：匹配由 x 指明的控制字符。例如，\cM 匹配一个 Control-M 或回车符。x 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的“c”字符。
- \d：匹配一个数字字符。等价于[0-9]。
- \D：匹配一个非数字字符。等价于[^0-9]。
- \f：匹配一个换页符。等价于\x0c 和\cL。
- \n：匹配一个换行符。等价于\x0a 和\cJ。
- \s：匹配任何不可见字符，包括空格、制表符、换页符等等。等价于[\f\n\r\t\v]。
- \S：匹配任何可见字符。等价于[^ \f\n\r\t\v]。
- \t：匹配一个制表符。等价于\x09 和\cI。
- \v：匹配一个垂直制表符。等价于\x0b 和\cK。
- \w：匹配包括下划线的任何单词字符。类似但不等价于“[A-Za-z0-9_]”，这里的“单词”字符使用 Unicode 字符集。
- \W：匹配任何非单词字符。等价于“[^A-Za-z0-9_]”。
- \xn：匹配 n，其中 n 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，“\x41”匹配“A”。“\x041”则等价于“\x04&1”。正则表达式中可以使用 ASCII 编码。
- \num：匹配 num，其中 num 是一个正整数。对所获取的匹配的引用。例如，“(.)\1”匹配两个连续的相同字符。
- \n：标识一个八进制转义值或一个向后引用。如果\n 之前至少 n 个获取的子表达式，则 n 为向后引用。否则，如果 n 为八进制数字（0-7），则 n 为一个八进制转义值。
- \nm：标识一个八进制转义值或一个向后引用。如果\nm 之前至少有 nm 个获得子表达式，则 nm 为向后引用。如果\nm 之前至少有 n 个获取，则 n 为一个后跟文字 m 的向后引用。

如果前面的条件都不满足，若 n 和 m 均为八进制数字（0-7），则\nm 将匹配八进制转义值 nm。

- \nml：如果 n 为八进制数字（0-7），且 m 和 l 均为八进制数字（0-7），则匹配八进制转义值 nml。
- \un：匹配 n，其中 n 是一个用 4 个十六进制数字表示的 Unicode 字符。例如，\u00A9 匹配版权符号（©）。
- \<\>：匹配词（word）的开始（\<）和结束（\>）。例如正则表达式\<the\>能够匹配字符串"for the wise"中的"the"，但是不能匹配字符串"otherwise"中的"the"。注意：这个元字符不是所有的软件都支持的。
- \(\)：将\(和\)之间的表达式定义为“组”（group），并且将匹配这个表达式的字符保存到一个临时区域（一个正则表达式中最多可以保存 9 个），它们可以用\1 到\9 的符号来引用。
- |：将两个匹配条件进行逻辑“或”（Or）运算。例如正则表达式（him|her）匹配"it belongs to him"和"it belongs to her"，但是不能匹配"it belongs to them."。注意：这个元字符不是所有的软件都支持的。
- +：匹配 1 或多个正好在它之前的那个字符。例如正则表达式 9+匹配 9、99、999 等。注意：这个元字符不是所有的软件都支持的。
- ?：匹配 0 或 1 个正好在它之前的那个字符。注意：这个元字符不是所有的软件都支持的。
- {i}{ij}：匹配指定数目的字符，这些字符是在它之前的表达式定义的。例如正则表达式 A[0-9]{3} 能够匹配字符"A"后面跟着正好 3 个数字字符的串，例如 A123、A348 等，但是不匹配 A1234。而正则表达式[0-9]{4,6} 匹配连续的任意 4 个、5 个或者 6 个数字。

再来看下简单的实例。

1. 验证用户名和密码

```
"^[a-zA-Z]\w{5,15}$"
```

正确格式："[A-Z][a-z]_[0-9]"组成，并且第 1 个字必须为字母 6~16 位。

2. 验证电话号码

```
"^(\\d{3,4}-)\\d{7,8}$"
```

正确格式：xxx/xxxx-xxxxxxx/xxxxxxxx

3. 验证手机号码

```
"^1[3|4|5|7|8][0-9]\\d{8}$"
```

4. 验证身份证号（15 位或 18 位数字）

```
"\\d{14}[[0-9],0-9xX]"
```

5. 验证 Email 地址

```
"^\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\.\\w+([-.]\\w+)*$"
```

6. 只能输入由数字和 26 个英文字母组成的字符串

```
"^[A-Za-z0-9]+$"
```

7. 整数或者小数

```
"^[0-9]+([.][0-9]+){0,1}$"
```

8. 只能输入数字

```
"^[0-9]*$"
```

9. 只能输入 n 位的数字

```
"^\\d{n}$"。
```

10. 只能输入至少 n 位的数字

```
"^\\d{n,}$"
```

11. 只能输入 m~n 位的数字

```
"^\\d{m,n}$"
```

12. 只能输入零和非零开头的数字

```
"^(0|[1-9][0-9]*)$"
```

13. 只能输入有两位小数的正实数

```
"^[0-9]+(.[0-9]{2})?$"
```

14. 只能输入有 1~3 位小数的正实数

```
"^[0-9]+(\.[0-9]{1,3})?$"
```

15. 只能输入非零的正整数

```
"^\+?[1-9][0-9]*$"
```

16. 只能输入非零的负整数

```
"^\-[1-9][0-9]*$"
```

17. 只能输入长度为 3 的字符

```
"^.{3}$"
```

18. 只能输入由 26 个英文字母组成的字符串

```
"^[A-Za-z]+$"
```

19. 只能输入由 26 个大写英文字母组成的字符串

```
"^[A-Z]+$"
```

20. 只能输入由 26 个小写英文字母组成的字符串

```
"^[a-z]+$"
```

21. 验证是否含有^%&',;=?$\"等字符

```
"[^%&',;=?$\x22]+"
```

22. 只能输入汉字

```
"^[\u4e00-\u9fa5]{0,}$"
```

23. 验证 URL

```
"^http://([\\w-]+\.)+[\\w-]+(/[\\w-./?%&=]*)?$"
```

24. 验证一年的 12 个月

```
"^(0?[1-9]|1[0-2])$"
正确格式为："01"～"09"和"10"～"12"。
```

25. 验证一个月的 31 天

```
"^((0?[1-9])|((1|2)[0-9])|30|31)$"
正确格式为："01"～"09"、"10"～"29"和"30"~"31"。
```

26. 获取日期正则表达式

```
"\\d{4}[年|\-|\.]\\d{\1-\12}[月|\-|\.]\\d{\1-\31}日? "
```



可用来匹配大多数年月日信息。

27. 匹配双字节字符（包括汉字在内）

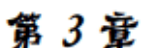
```
" [^\x00-\xff] "
```

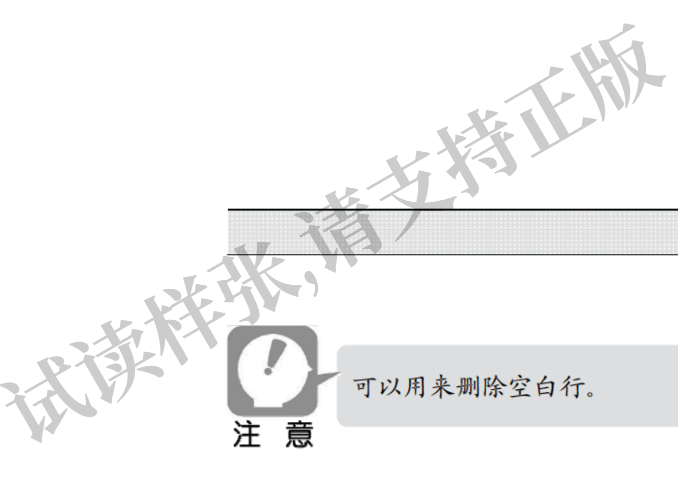


可以用来计算字符串的长度（一个双字节字符长度计 2，ASCII 字符计 1）。

28. 匹配空白行的正则表达式

```
"\n\s*\r"
```





注 意

可以用来删除空白行。

29. 匹配 HTML 标记的正则表达式

```
"<（\S*?）[^>]*>.*?</>|<.*? />"
```



注 意

网上流传的版本太糟糕，上面这个也仅仅能匹配部分，对于复杂的嵌套标记依旧无能为力。

30. 匹配首尾空白字符的正则表达式

```
"^\s*|\s*$"
```



注 意

可以用来删除行首行尾的空白字符（包括空格、制表符、换页符等等），非常有用的表达式。

31. 匹配网址 URL 的正则表达式

```
" [a-zA-z]+://[^\s]* "
```



注 意

网上流传的版本功能很有限，上面这个基本可以满足需求。

32. 匹配账号是否合法（字母开头，允许 5~16B，允许字母数字下划线）

```
"^[a-zA-Z][a-zA-Z0-9_]{4,15}$"
```



注 意

表单验证时很实用。

33. 匹配腾讯 QQ 号

```
" [1-9][0-9]{4,} "
```



注 意

腾讯 QQ 号从 10 000 开始。

34. 匹配中国邮政编码

```
" [1-9]\\d{5}（?!\d）"
```


注 意

中国邮政编码为 6 位数字。

35. 匹配 ip 地址

```
"([1-9]{1,3}\.){3}[1-9]"
```


注 意

提取 ip 地址时有用。

36. 匹配 MAC 地址

```
"([A-Fa-f0-9]{2}\:){5}[A-Fa-f0-9] "
```

正则表达式的应用范围很广，sed、awk、Python、PHP、Perl……都支持正则表达式。熟练掌握正则后，使用 Linux 时会方便很多。

3.5.2 grep

Linux 系统中 grep 命令是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹配的行打印出来。grep 全称是 Global Regular Expression Print，表示全局正则表达式版本，它的使用权限是所有用户。

UNIX 的 grep 家族包括 grep、egrep 和 fgrep。egrep 和 fgrep 的命令只跟 grep 有很小差异。egrep 是 grep 的扩展，支持更多的 re 元字符， fgrep 就是 fixed grep 或 fast grep，它们把所有的字母都看作单词，也就是说，正则表达式中的元字符表示其自身的字面意义，不再特殊。Linux 使用 GNU 版本的 grep。它功能更强，可以通过-G、-E、-F 命令行选项来使用 egrep 和 fgrep 的功能。

1. 参数简介

先来看下 man grep 是怎么说的吧，如图 3-17 所示。

```
pi@raspberrypi: ~/code/python/hello
GREP(1)                                                        GREP(1)

NAME
       grep, egrep, fgrep, rgrep - print lines matching a pattern

SYNOPSIS
       grep [OPTIONS] PATTERN [FILE...]
       grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION
       grep  searches the named input FILEs (or standard input if no files are
       named, or if a single hyphen-minus (-) is given as file name) for lines
       containing  a  match to the given PATTERN.  By default, grep prints the
       matching lines.

       In  addition,  three  variant  programs  egrep,  fgrep  and  rgrep  are
       available.   egrep  is  the  same  as  grep -E.   fgrep  is the same as
       grep -F.  rgrep is the same as grep -r.  Direct  invocation  as  either
       egrep  or  fgrep  is  deprecated,  but  is provided to allow historical
       applications that rely on them to run unmodified.
```

图 3-17　man grep

grep 命令的参数

- -a：将 binary 文件以 text 文件的方式搜寻数据；
- -c：计算找到 '搜寻字符串' 的次数；
- -i：忽略大小写的不同，所以大小写视为相同；
- -n：顺便输出行号；
- -v：反向选择，亦即显示出没有"搜寻字符串"内容的那一行；
- --color=auto：可以将找到的关键词部分加上颜色显示；
- -E：也就是 egrep，grep 的基础上扩展了正则功能；
- -F：也就是 fgpre，将所有的字符当成元字符来过滤，不识别正则。因此速度较快。

2. 实例测试

grep –c 测试，如图 3-18 所示。

```
pi@raspberrypi: ~/code/python/getNip
pi@raspberrypi ~ $ cd code/python/getNip/
pi@raspberrypi ~/code/python/getNip $ ls -l
total 4
-rw-r--r-- 1 pi pi 861 Aug 28 10:32 getNip.py
pi@raspberrypi ~/code/python/getNip $ grep -c print getNip.py
2
pi@raspberrypi ~/code/python/getNip $ 
```

图 3-18　grep –c 测试

grep -i 测试，如图 3-19 所示。

```
pi@raspberrypi: ~/code/python/getNip
pi@raspberrypi ~ $ cd code/python/getNip/
pi@raspberrypi ~/code/python/getNip $ ls -l
total 4
-rw-r--r-- 1 pi pi 861 Aug 28 10:32 getNip.py
pi@raspberrypi ~/code/python/getNip $ grep -i IMPORT getNip.py
import urllib2
import re
import os
pi@raspberrypi ~/code/python/getNip $ 
```

图 3-19　grep –i 测试

grep -n 测试，这个在使用 vi 敲代码的时候特别有用，如图 3-20 所示。

```
pi@raspberrypi: ~/code/python/getNip
pi@raspberrypi ~ $ cd code/python/getNip/
pi@raspberrypi ~/code/python/getNip $ ls -l
total 4
-rw-r--r-- 1 pi pi 861 Aug 28 10:32 getNip.py
pi@raspberrypi ~/code/python/getNip $ grep -n print getNip.py
28:                     print("Nip = %s"%self.Nip)
30:                     print("未取得本机NIP")
pi@raspberrypi ~/code/python/getNip $ 
```

图 3-20　grep –n 测试

grep –v 测试，查找不含()的非空行，如图 3-21 所示。

```
pi@raspberrypi ~ $ cd code/python/getNip/
pi@raspberrypi ~/code/python/getNip $ ls -l
total 4
-rw-r--r-- 1 pi pi 861 Aug 28 10:32 getNip.py
pi@raspberrypi ~/code/python/getNip $ grep -v "(.*)" getNip.py | grep -v "^$"
#!/usr/bin/env python
# -*- coding:utf-8 -*-
#Author  :hstking
#E-mail  :hstking@hotmail.com
#Ctime   :2015/08/27
#Mtime   :
#Version :
import urllib2
import re
import os
                self.nipFile = self.logPath + os.sep + 'Nip.txt'
                self.nip = None
                urls = 'http://1111.ip138.com/ic.asp'
                else:
                        pass
                else:
if __name__ == '__main__':
pi@raspberrypi ~/code/python/getNip $
```

图 3-21　grep –v 测试

grep –E 测试，grep 查询时包含正则，如图 3-22 所示。

```
pi@raspberrypi ~/code/python/getNip $ grep -n -E "pass|paSS" getNip.py
34:                     pass
pi@raspberrypi ~/code/python/getNip $
```

图 3-22　grep –E 测试

grep –F 测试，grep 查询时当成元字符，如图 3-23 所示。

```
pi@raspberrypi ~/code/python/getNip $ grep -F "." getNip.py
#E-mail  :hstking@hotmail.com
                self.logPath = os.path.expanduser('~') + os.sep + 'log'
                self.nipFile = self.logPath + os.sep + 'Nip.txt'
                self.Nip = None
                self.getNip()
                self.writeNip()
                urls = 'http://1111.ip138.com/ic.asp'
                if urllib2.urlopen(urls).geturl() == urls:
                        rawString = urllib2.urlopen(urls).read()
                        self.Nip = re.search(b'\d+\.\d+\.\d+\.\d+',rawString).gr
oup()
                        print("Nip = %s"%self.Nip)
                if os.path.isdir(self.logPath):
                        os.makedirs(self.logPath)
                with open(self.nipFile,'w') as FP:
                        FP.write(self.Nip)
pi@raspberrypi ~/code/python/getNip $
```

图 3-23　grep –F 测试

grep 功能强大，在文本内搜索效率非常高，可以说是 Linux 中最常用的几个命令。

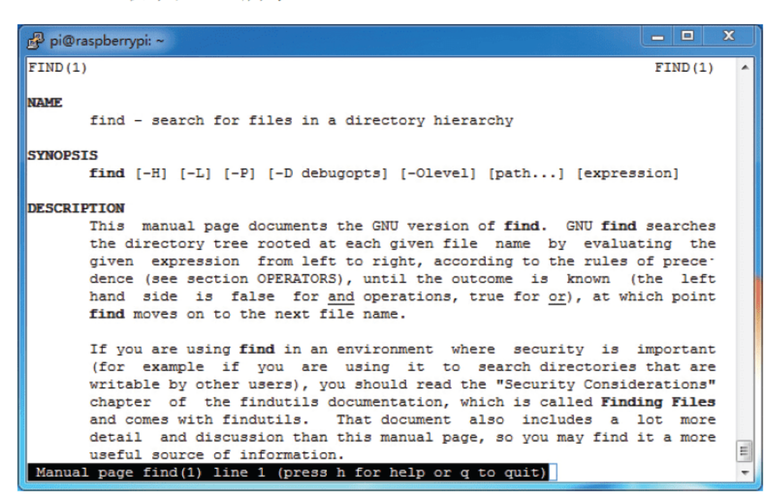
3.5.3　find

Linux 下 find 命令提供了相当多的查找条件，功能很强大。find 命令在目录结构中搜索文件，并执行指定的操作。由于 find 具有强大的功能，所以它的选项也很多，其中大部分选项都值得我们花时间来了解一下。即使系统中含有网络文件系统（NFS），find 命令在该文件系统中同样有效，具有相应的权限。 在运行一个非常消耗资源的 find 命令时，很多人都倾向于把它放在后台执行，

因为遍历一个大的文件系统可能会花费很长的时间（这里是指 30GB 以上的文件系统）。

1. 参数简介

先来看下 man find，如图 3-24 所示。


```
FIND(1)                                                          FIND(1)

NAME
       find - search for files in a directory hierarchy

SYNOPSIS
       find [-H] [-L] [-P] [-D debugopts] [-Olevel] [path...] [expression]

DESCRIPTION
       This  manual page documents the GNU version of find.  GNU find searches
       the directory tree rooted at each given file  name  by  evaluating  the
       given  expression  from left to right, according to the rules of prece-
       dence (see section OPERATORS), until the outcome  is  known  (the  left
       hand  side  is  false  for and operations, true for or), at which point
       find moves on to the next file name.

       If you are using find in an environment  where  security  is  important
       (for  example  if  you  are  using  it  to  search directories that are
       writable by other users), you should read the "Security Considerations"
       chapter  of  the findutils documentation, which is called Finding Files
       and comes with findutils.   That document  also  includes  a  lot  more
       detail  and discussion than this manual page, so you may find it a more
       useful source of information.
 Manual page find(1) line 1 (press h for help or q to quit)
```

图 3-24 man find

Linux 中 find 常见用法：

```
find  path  -option  [  -print  ]  [ -exec  -ok  command  ]  {} \;
```

find 命令的参数：

- pathname find：所查找的目录路径。例如用.来表示当前目录，用/来表示系统根目录。
- -print find：将匹配的文件输出到标准输出。
- -exec find：对匹配的文件执行该参数所给出的 shell 命令。相应命令的形式为'command' { } \;，注意{ }和\;之间的空格。



注意

在使用 find 命令的-exec 选项处理匹配到的文件时，find 命令将所有匹配到的文件一起传递给 exec 执行。不幸的是，有些系统对能够传递给 exec 的命令长度有限制，这样在 find 命令运行几分钟之后，就会出现溢出错误。错误信息通常是“参数列太长”或“参数列溢出”。此时就该 xargs 命令大显身手了，xargs 与 find 命令一起使用。find 命令把匹配到的文件传递给 xargs 命令，而 xargs 命令每次只获取一部分文件而不是全部，不像- exec 选项那样。这样它可以先处理最先获取的一部分文件，然后是下一批，并如此继续下去。在有些系统中，使用-exec 选项会为处理每一个匹配到的文件而发起一个相应的进程，并非将匹配到的文件全部作为参数一次执行；这样在有些情况下就会出现进程过多，系统性能下降的问题，因而效率不高；而使用 xargs 命令则只有一个进程。另外，在使用 xargs 命令时，究竟是一次获取所有的参数，还是分批取得参数，以及每一次获取参数的数目都会根据该命令的选项及系统内核中相应的可调参数来确定。

- -ok 和-exec: 作用相同，只不过以一种更为安全的模式来执行该参数所给出的 shell 命令，在执行每一个命令之前，都会给出提示，让用户来确定是否执行。
- -name filename：查找名为 filename 的文件。
- -perm：按执行权限来查找。
- -user username：按文件属主来查找。
- -group groupname：按组来查找。
- -mtime -n +n：按文件更改时间来查找文件，-n 指 n 天以内，+n 指 n 天以前。
- -atime -n +n：按文件访问时间来查 GIN: 0px">。
- -ctime -n +n：按文件创建时间来查找文件，-n 指 n 天以内，+n 指 n 天以前。
- -nogroup：查找无有效属组的文件，即文件的属组在/etc/groups 中不存在。
- -nouser：查找无有效属主的文件，即文件的属主在/etc/passwd 中不存在。
- -newer f1 !f2：查找文件，-n 指 n 天以内，+n 指 n 天以前。
- -type b/d/c/p/l/f ：查找块设备、目录、字符设备、管道、符号链接、普通文件。
- -size n[c]：查找长度为 n 块[或 n 字节]的文件。
- -depth：使查找在进入子目录前先行查找完本目录。
- -mount：查找文件时不跨越文件系统 mount 点。
- -follow：如果遇到符号链接文件，就跟踪链接所指的文件。
- -cpio %;：查找位于某一类型文件系统中的文件，这些文件系统类型通常可在/etc/fstab 中找到。
- -prune：忽略某个目录。

这么多的参数看上去是不是很复杂？实际上常用的参数并不多，最常用的就是-name 了。参考下文的几个实例就明白了，其实 find 很简单。

2. 实例测试

这里只展示几个最常用的 find 用法，如图 3-25 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo find /etc -name passwd
/etc/pam.d/passwd
/etc/cron.daily/passwd
/etc/passwd
pi@raspberrypi ~ $ sudo find /etc -name passwd
/etc/pam.d/passwd
/etc/cron.daily/passwd
/etc/passwd
pi@raspberrypi ~ $ sudo find /etc -size +100k
/etc/ImageMagick/mime.xml
/etc/ssh/moduli
/etc/ssl/certs/ca-certificates.crt
pi@raspberrypi ~ $ sudo find /etc -size +100k -exec ls -lh {} \;
-rw-r--r-- 1 root root 131K Mar  9  2014 /etc/ImageMagick/mime.xml
-rw-r--r-- 1 root root 133K Jul 12  2014 /etc/ssh/moduli
-rw-r--r-- 1 root root 267K May  7 06:34 /etc/ssl/certs/ca-certificates.crt
pi@raspberrypi ~ $
```

图 3-25 find 实例

find 用于查找符合条件的文件，效率比 Windows 下的“查找”高很多，功能也远比“查找”强大。



注 意

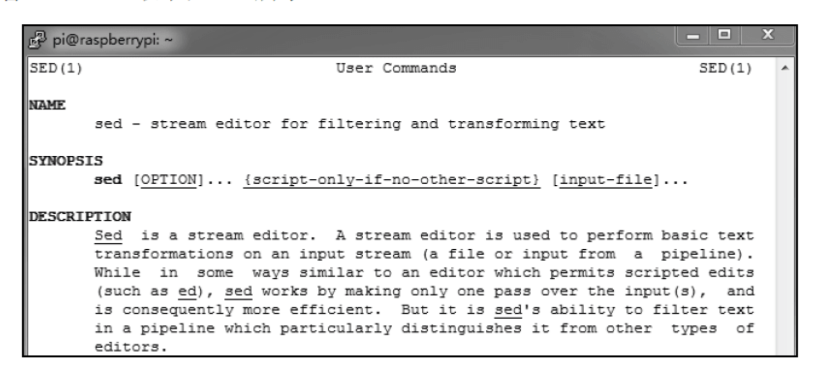
find 有独特的 exec 参数可以配合处理查找结果（也可以配合 xargs 命令）。有时 Linux 的日志文件或是其他的文件在未做限制的情况下会增长到让人瞠目结舌，这时候就是 find 大显身手的时候，只需一个 find 就可以让系统恢复正常，非常方便。

3.5.4 sed

sed 是一种在线编辑器，它一次处理一行内容。处理时，把当前处理的行存储在临时缓冲区中，称为“模式空间”（pattern space），接着用 sed 命令处理缓冲区中的内容，处理完成后，把缓冲区的内容送往屏幕。接着处理下一行，这样不断重复，直到文件末尾。文件内容并没有改变，除非使用重定向存储输出。sed 主要用来自动编辑一个或多个文件；简化对文件的反复操作；编写转换程序等。

1. 参数简介

先来看看 man sed，如图 3-26 所示。



```
pi@raspberrypi: ~
SED(1)                        User Commands                        SED(1)

NAME
       sed - stream editor for filtering and transforming text

SYNOPSIS
       sed [OPTION]... {script-only-if-no-other-script} [input-file]...

DESCRIPTION
       Sed  is a stream editor.  A stream editor is used to perform basic text
       transformations on an input stream (a file or input from  a  pipeline).
       While  in  some  ways similar to an editor which permits scripted edits
       (such as ed), sed works by making only one pass over the input(s),  and
       is consequently more efficient.  But it is sed's ability to filter text
       in a pipeline which particularly distinguishes it from other  types  of
       editors.
```

图 3-26 man sed

下面是 sed 命令的参数的常用选项：

- -n：使用安静（silent）模式。在一般 sed 的用法中，所有来自 STDIN 的资料一般都会被列出到屏幕上。但如果加上-n 参数后，则只有经过 sed 特殊处理的那一行（或者动作）才会被列出来。
- -e：直接在指令列模式上进行 sed 的动作编辑。
- -f：直接将 sed 的动作写在一个档案内，-f filename 则可以执行 filename 内的 sed 动作。
- -r：sed 的动作支援的是延伸型正规表示法的语法。（预设是基础正规表示法语法）。
- -i：直接修改读取的档案内容，而不是由屏幕输出。

常用命令：

- a：新增，a 的后面可以接字串，而这些字串会在新的一行出现（目前的下一行）。
- c：取代，c 的后面可以接字串，这些字串可以取代 n1 与 n2 之间的行。
- d：删除，因为是删除，所以 d 后面通常不接任何内容。

- i：插入，i 的后面可以接字符串，而这些字符串会在新的一行出现（目前的上一行）。
- p：列印，即将某个选择的资料印出。通常 p 会与参数 sed -n 一起运作。
- s：取代，可以直接进行取代的工作。通常这个 s 的动作可以搭配正规表示法。

2. 创建测试环境

下面来实战演练一下，先建立一个演示用的文本，执行命令：

```
echo "aaaa" > /tmp/test.txt
echo "bbbb" >> /tmp/test.txt
echo "cccc" >> /tmp/test.txt
echo "1111" >> /tmp/test.txt
echo "2222" >> /tmp/test.txt
echo "3333" >> /tmp/test.txt
```

建立好演示用文本/tmp/test.txt，如图 3-27 所示。

```
pi@raspberrypi ~ $ echo "aaaa" > /tmp/test.txt
pi@raspberrypi ~ $ echo "bbbb" >> /tmp/test.txt
pi@raspberrypi ~ $ echo "cccc" >> /tmp/test.txt
pi@raspberrypi ~ $ echo "1111" >> /tmp/test.txt
pi@raspberrypi ~ $ echo "2222" >> /tmp/test.txt
pi@raspberrypi ~ $ echo "3333" >> /tmp/test.txt
pi@raspberrypi ~ $ cat /tmp/test.txt
aaaa
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $
```

图 3-27　创建演示文本

3. 实例测试

使用 sed 删除功能，如图 3-28 所示。

```
pi@raspberrypi ~ $ sed '1d' /tmp/test.txt
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed '$d' /tmp/test.txt
aaaa
bbbb
cccc
1111
2222
pi@raspberrypi ~ $ sed '1,2d' /tmp/test.txt
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed '2,$d' /tmp/test.txt
aaaa
pi@raspberrypi ~ $
```

图 3-28　sed 删除



注意

sed 只有在使用-i 参数时才会对文件进行真正的修改。

使用 sed 搜索或替换特定字符（如果仅仅是搜索，这个功能与 grep 很相似），如图 3-29 所示。

```
pi@raspberrypi ~ $ sed -n '/1111/p' /tmp/test.txt
1111
pi@raspberrypi ~ $ sed 's/1111/0000/g' /tmp/test.txt
aaaa
bbbb
cccc
0000
2222
3333
pi@raspberrypi ~ $
```

图 3-29　sed 搜索、替换

使用 sed 添加字符或添加行，如图 3-30 所示。

```
pi@raspberrypi ~ $ sed 's/aaaa/&xxxx/g' /tmp/test.txt
aaaaxxxx
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed '/aaaa/a xxxx' /tmp/test.txt
aaaa
xxxx
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed '1a xxxx' /tmp/test.txt
aaaa
xxxx
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $
```

图 3-30　sed 添加字符、行

使用 sed 删除字符或删除行，如图 3-31 所示。

```
pi@raspberrypi ~ $ sed 's/aaaa//g' /tmp/test.txt

bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed '/aaaa/d' /tmp/test.txt
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed '1d' /tmp/test.txt
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $
```

图 3-31　sed 删除字符、行

使用 sed –i 直接修改文件，如图 3-32 所示。

```
pi@raspberrypi ~ $ cat /tmp/test.txt
aaaa
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed '1,5d' /tmp/test.txt
3333
pi@raspberrypi ~ $ cat /tmp/test.txt
aaaa
bbbb
cccc
1111
2222
3333
pi@raspberrypi ~ $ sed -i '1,5d' /tmp/test.txt
pi@raspberrypi ~ $ cat /tmp/test.txt
3333
pi@raspberrypi ~ $
```

图 3-32　sed –i 直接修改文件

与 vim 不同的是 vim 需要打开文件才能编辑内容，而 sed 命令配合-i 选项可以不打开文件直接编辑。虽然 vim 也有查找、搜索、替换的功能，但少一道工序不是更好吗？除了以上展示的例子外，sed 还有很多强大的功能待挖掘。熟悉了 sed，在以后的文本编辑中会方便很多。



注　意

大部分情况下，sed 脚本无论多长都能写成单行的形式（通过`-e'选项和`;'号）——只要命令解释器支持，所以这里说的单行脚本除了能写成一行还对长度有所限制。因为这些单行脚本的意义不在于它们是以单行的形式出现，而是让用户能方便地在命令行中使用这些紧凑的脚本。

3.5.5　awk

awk（其名称得自于它的创始人 Alfred Aho 、Peter Weinberger 和 Brian Kernighan 姓氏的首个字母）是一种优良的文本处理工具。它不仅是 Linux 中，也是任何环境中，现有的功能最强大的数据处理引擎之一。awk 提供了极其强大的功能：可以进行样式装入、流控制、数学运算符、进程控制语句甚至于内置的变量和函数。它具备了一个完整的语言所应具有的几乎所有精美特性。实际上 awk 的确拥有自己的语言：awk 程序设计语言，3 位创建者已将它正式定义为“样式扫描和处理语言”。它允许创建简短的程序，这些程序读取输入文件、为数据排序、处理数据、对输入执行计算以及生成报表，还有无数其他的功能。

最简单地说，awk 是一种用于处理文本的编程语言工具。awk 在很多方面类似于 shell 编程语言，尽管 awk 具有完全属于其本身的语法。它的设计思想来源于 SNOBOL4、sed、Marc Rochkind 设计的有效性语言以及语言工具 yacc 和 lex，当然还从 C 语言中获取了一些优秀的思想。在最初创造 awk 时，其目的是用于文本处理，并且这种语言的基础是，只要在输入数据中有模式匹配，就执行一系列指令。该实用工具扫描文件中的每一行，查找与命令行中所给定内容相匹配的模式。如果发现匹配内容，则进行下一个编程步骤。如果找不到匹配内容，则继续处理下一行。

1. 参数简介

先来看看 man awk，如图 3-33 所示。

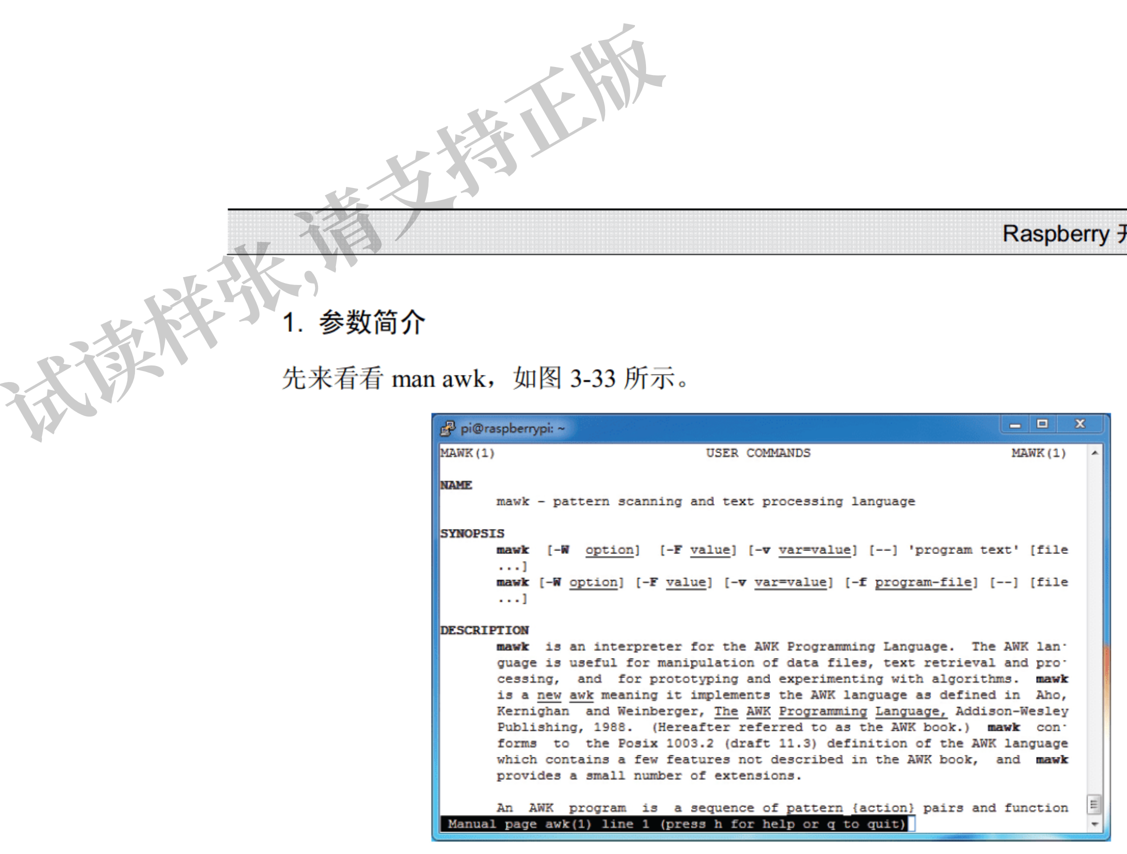


图 3-33　man awk

有 3 种方式调用 awk：

- 命令行方式

```
awk [-F  field-separator]  'commands'  input-file(s)
```

其中，commands 是真正 awk 命令，[-F 域分隔符]是可选的。input-file（s）是待处理的文件。

在 awk 文件的每一行中，由域分隔符分开的每一项称为一个域。通常，在不指名-F 域分隔符的情况下，默认的域分隔符是空格。

- shell 脚本方式

将所有的 awk 命令插入一个文件，并使 awk 程序可执行，然后 awk 命令解释器作为脚本的首行，通过键入脚本名称来调用。

相当于 shell 脚本首行的：#!/bin/sh

可以换成：#!/bin/awk

- 将所有的 awk 命令插入一个单独文件，然后调用：

```
awk -f awk-script-file input-file(s)
```

其中，-f 选项加载 awk-script-file 中的 awk 脚本，input-file（s）跟上面的是一样的。

在这里只介绍命令行方式，如对其他方式有兴趣，请自行参考谷歌百度。

awk 有许多内置变量用来设置环境信息，这些变量可以被改变，下面给出了最常用的一些变量。

```
ARGC 命令行变元个数
ARGV 命令行变元数组
FILENAME 当前输入文件名
FNR 当前文件中的记录号
```

```
FS 输入域分隔符，默认为一个空格
RS 输入记录分隔符
NF 当前记录里域个数
NR 到目前为止记录数
OFS 输出域分隔符
ORS 输出记录分隔符
```

2. 创建测试环境

下面来实战演练一下，先建立一个演示用的文本，执行命令：

```
echo "aaa bbb:ccc ddd" > /tmp/test.txt
echo "111 222:333 444" >> /tmp/test.txt
echo "AAA BBB:CCC DDD" >> /tmp/test.txt
echo "[[[ ]]]:{{{ }}}" >> /tmp/test.txt
```

建立好演示用文本/tmp/test.txt，如图 3-34 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ cat /tmp/test.txt
aaa bbb:ccc ddd
111 222:333 444
AAA BBB:CCC DDD
[[[ ]]]:{{{ }}}
pi@raspberrypi ~ $
```

图 3-34　演示用文本

3. 实例测试

使用 awk 显示第 2~4 行，如图 3-35 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ awk 'NR==2,NR==4 {print}' /tmp/test.txt
111 222:333 444
AAA BBB:CCC DDD
[[[ ]]]:{{{ }}}
pi@raspberrypi ~ $
```

图 3-35　awk 显示指定行

使用 awk 演示输入\输出记录分隔符，如图 3-36 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ awk 'BEGIN{RS=":"} {print}' /tmp/test.txt
aaa bbb
ccc ddd
111 222
333 444
AAA BBB
CCC DDD
[[[ ]]]
{{{ }}}

pi@raspberrypi ~ $ awk 'BEGIN{RS=":"}{ORS="aa\n"} {print}' /tmp/test.txt
aaa bbbaa
ccc ddd
111 222aa
333 444
AAA BBBaa
CCC DDD
[[[ ]]]aa
{{{ }}}
aa
pi@raspberrypi ~ $
```

图 3-36　awk 记录分隔符

使用 awk 演示输入\输出域分隔符，如图 3-37 所示。

```
pi@raspberrypi ~ $ awk  '{print $1}' /tmp/test.txt
aaa
111
AAA
[[[
pi@raspberrypi ~ $ awk  -F ":" '{print $1}' /tmp/test.txt
aaa bbb
111 222
AAA BBB
[[[ ]]]
pi@raspberrypi ~ $ awk 'BEGIN{OFS="\t"} {print $1,$2}' /tmp/test.txt
aaa     bbb:ccc
111     222:333
AAA     BBB:CCC
[[[     ]]]:{{{
pi@raspberrypi ~ $ awk -F ":" 'BEGIN{OFS="\t"} {print $1,$2}' /tmp/test.txt
aaa bbb ccc ddd
111 222 333 444
AAA BBB CCC DDD
[[[ ]]] {{{ }}}
pi@raspberrypi ~ $
```

图 3-37　awk 域分隔符

使用 awk 演示 NR、NF、FNR，如图 3-38 所示。

```
pi@raspberrypi ~ $ awk '{print FNR}' /tmp/test.txt
1
2
3
4
pi@raspberrypi ~ $ awk '{print NR}' /tmp/test.txt
1
2
3
4
pi@raspberrypi ~ $ awk '{print NF}' /tmp/test.txt
3
3
3
3
pi@raspberrypi ~ $ awk -F ":" '{print NF}' /tmp/test.txt
2
2
2
2
pi@raspberrypi ~ $
```

图 3-38　awk 演示内置变量

awk 和 sed 都是用来处理文本文件的，它们的用途各有侧重点，也有部分功能重合。个人认为 awk 在处理有规律的文本时更加方便。还有更多功能待发掘。这里只是简单介绍，如有兴趣可参考谷歌百度。



注　意

awk 功能极其强大的，它可以进行样式装入、流控制、数学运算符、进程控制语句甚至于内置的变量和函数。 具备了一个完整的语言所应具有的几乎所有精美特性。实际上，awk 的确拥有自己的语言：awk 程序设计语言，awk 的 3 位创建者已将它正式定义为样式扫描和处理语言。

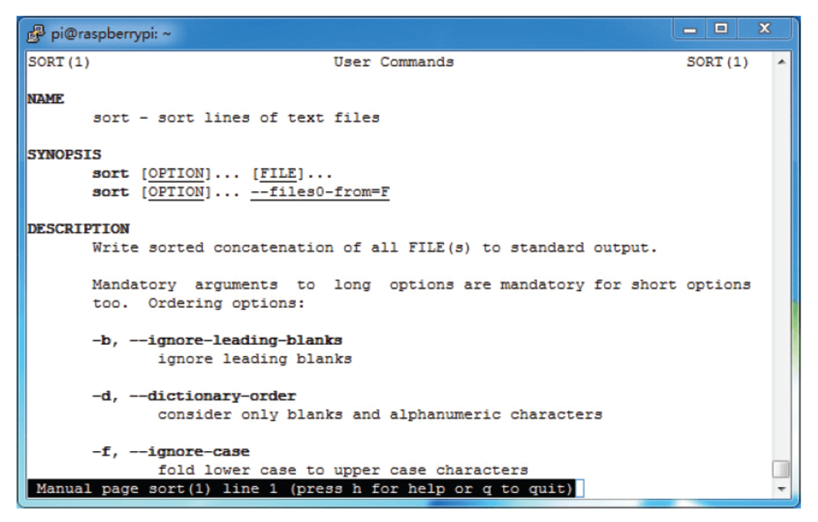
3.5.6 其他常用工具

1. sort

sort 顾名思义，这个命令就是用来排序的。

（1）参数简介

首先还是看看 man sort，如图 3-39 所示。



```
SORT(1)                          User Commands                          SORT(1)

NAME
       sort - sort lines of text files

SYNOPSIS
       sort [OPTION]... [FILE]...
       sort [OPTION]... --files0-from=F

DESCRIPTION
       Write sorted concatenation of all FILE(s) to standard output.

       Mandatory  arguments  to  long  options are mandatory for short options
       too.  Ordering options:

       -b, --ignore-leading-blanks
              ignore leading blanks

       -d, --dictionary-order
              consider only blanks and alphanumeric characters

       -f, --ignore-case
              fold lower case to upper case characters
 Manual page sort(1) line 1 (press h for help or q to quit)
```

图 3-39　man sort

sort 命令的常用参数如下：

- -f：忽略大小写的差异，例如 A 与 a 视为编码相同；
- -b：忽略最前面的空格符部分；
- -M：以月份的名字来排序，例如 JAN、DEC 等等的排序方法；
- -n：使用“纯数字”进行排序（默认是以文字型态来排序的）；
- -r：反向排序；
- -u：就是 uniq，相同的数据中，仅出现一行代表；
- -t：分隔符，默认是用 Tab 键来分隔；
- -k：以哪个区间（field）来进行排序的意思。

（2）创建测试环境

下面来看具体的实例。先构建测试文件/tmp/1.txt，如图 3-40 所示，执行命令：

```
echo "1 2 3 4:d" > /tmp/1.txt
echo "2 3 4 1:C" >> /tmp/1.txt
echo "3 4 1 2:a" >> /tmp/1.txt
echo "4 1 2 3:B" >> /tmp/1.txt
cat /tmp/1.txt
```




```
pi@raspberrypi ~ $ echo "1 2 3 4:d" > /tmp/1.txt
pi@raspberrypi ~ $ echo "2 3 4 1:C" >> /tmp/1.txt
pi@raspberrypi ~ $ echo "3 4 1 2:a" >> /tmp/1.txt
pi@raspberrypi ~ $ echo "4 1 2 3:B" >> /tmp/1.txt
pi@raspberrypi ~ $ cat /tmp/1.txt
1 2 3 4:d
2 3 4 1:C
3 4 1 2:a
4 1 2 3:B
pi@raspberrypi ~ $
```

图 3-40　实例文件

（3）实例测试

测试 sort –k 命令，如图 3-41。

```
pi@raspberrypi ~ $ sort -k 1 /tmp/1.txt
1 2 3 4:d
2 3 4 1:C
3 4 1 2:a
4 1 2 3:B
pi@raspberrypi ~ $ sort -k 4 /tmp/1.txt
2 3 4 1:C
3 4 1 2:a
4 1 2 3:B
1 2 3 4:d
pi@raspberrypi ~ $
```

图 3-41　sort –k 测试

测试 sort –r 命令，如图 3-42 所示。

```
pi@raspberrypi ~ $ sort -k 1 -r /tmp/1.txt
4 1 2 3:B
3 4 1 2:a
2 3 4 1:C
1 2 3 4:d
pi@raspberrypi ~ $ sort -k 4 -r /tmp/1.txt
1 2 3 4:d
4 1 2 3:B
3 4 1 2:a
2 3 4 1:C
pi@raspberrypi ~ $
```

图 3-42　sed –r 测试

测试 sort –t 命令，如图 3-43 所示。

```
pi@raspberrypi ~ $ sort -t ":" -k 2 /tmp/1.txt
3 4 1 2:a
4 1 2 3:B
2 3 4 1:C
1 2 3 4:d
pi@raspberrypi ~ $
```

图 3-43　sed –t 测试

在文本处理中，排序是很常见的。sort 命令也是必须掌握的命令。

2. uniq

uniq 命令可以去除排序过的文件中的重复行，因此 uniq 经常和 sort 合用。也就是说，为了使 uniq 起作用，所有的重复行必须是相邻的。

（1）参数简介

先看看 man uniq，如图 3-44 所示。

```
pi@raspberrypi: ~
UNIQ(1)                        User Commands                        UNIQ(1)

NAME
       uniq - report or omit repeated lines

SYNOPSIS
       uniq [OPTION]... [INPUT [OUTPUT]]

DESCRIPTION
       Filter  adjacent matching lines from INPUT (or standard input), writing
       to OUTPUT (or standard output).

       With no options, matching lines are merged to the first occurrence.

       Mandatory arguments to long options are  mandatory  for  short  options
       too.

       -c, --count
              prefix lines by the number of occurrences

       -d, --repeated
              only print duplicate lines

 Manual page uniq(1) line 1 (press h for help or q to quit)
```

图 3-44　man uniq

uniq 常用的选项如下：

- -i：忽略大小写字符的不同。
- -c：进行计数。
- -u：只显示唯一的行。

（2）创建测试环境

建立测试文件，执行命令：

```
echo "hello" > /tmp/1.txt
echo "HELLO" > /tmp/1.txt
echo "####" >> /tmp/1.txt
echo "aaaa" >> /tmp/1.txt
echo "aaaa" >> /tmp/1.txt
```

效果如图 3-45 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ echo "hello" > /tmp/1.txt
pi@raspberrypi ~ $ echo "HELLO" >> /tmp/1.txt
pi@raspberrypi ~ $ echo "####" >> /tmp/1.txt
pi@raspberrypi ~ $ echo "aaaa" >> /tmp/1.txt
pi@raspberrypi ~ $ echo "aaaa" >> /tmp/1.txt
pi@raspberrypi ~ $ cat /tmp/1.txt
hello
HELLO
####
aaaa
aaaa
pi@raspberrypi ~ $
```

图 3-45　实例文件

（3）实例测试

测试 uniq，如图 3-46 所示。

```
pi@raspberrypi ~ $ uniq /tmp/1.txt
hello
HELLO
####
aaaa
pi@raspberrypi ~ $
```

图 3-46 uniq 测试

测试 uniq –i，如图 3-47 所示。

```
pi@raspberrypi ~ $ uniq -i /tmp/1.txt
hello
####
aaaa
pi@raspberrypi ~ $
```

图 3-47 uniq –i 测试

测试 uniq –c，如图 3-48 所示。

```
pi@raspberrypi ~ $ uniq -c /tmp/1.txt
      1 hello
      1 HELLO
      1 ####
      2 aaaa
pi@raspberrypi ~ $
```

图 3-48 uniq –c 测试

uniq 一般和 sort 配合使用，先用 sort 排序，uniq 去除相邻的重复行。功能强大，速度快，非常方便。



杀鸡无须用牛刀，当 uniq+sort 能满足要求的时候就不要麻烦 awk 了。而且在处理小文本时，uniq+sort 的效率更高。

第 4 章

◂ Raspberry常用服务 ▸

想不想拥有一台个人的服务器？Raspberry 完全可以做到，它能完善绿色环保私人定制，一旦拥有别无所求。Raspberry 的操作系统 RaspBian 和 Debian 几乎没有什么区别。所以在 Debian 上有的服务在 RaspBian 上几乎都有。本章讲解在 RaspBian 上安装配置 Linux 最常用的服务。

本章主要内容包括：

- xrdp 远程桌面服务
- samba 共享服务
- miniDLNA 共享影音服务
- vsftp FTP 服务
- Nginx 和 LAMP 服务

4.1 xrdp 远程桌面服务

计算机组成网络，第一步通常是远程连接。紧接着的一定是远程控制了。远程控制的明星软件很多，比如最出名的 TeamViewer。那的确是非常好的软件，即使在 Linux 下表现同样出色，可是杀鸡何须用牛刀。也许还有更合适的选择。

4.1.1 xrdp 简介

远程桌面不管是在 Linux 还是 Widows，使用都非常频繁。Xrdp 是 Linux 的远程桌面服务，一般常用的 Linux 远程桌面是 VNC，但是在 Windows 上连接 Linux 的 VNC 还得下载专用的工具。VNC 的服务端口是 5900+x，与常用的 Windows 远程桌面端口 3389 不一样。所以还是选择 xrdp 服务吧，实际上 xrdp 服务也是依赖 VNC 服务的。

4.1.2 xrdp 安装

在 RaspBian 上安装 xrdp 服务非常简单，用 Putty 以默认用户 pi 登录到 Raspberry 后，执行命令：

```
sudo apt-get install xrdp
```

安装完毕后，也无须配置，只要每次使用前启动 xrdp 服务就可以了。执行命令：

```
sudo /etc/init.d/xrdp start
sudo update-rc.d xrdp defaults
```

第一条命令是启动 xrdp 服务，第二条命令是将 xrdp 服务加入系统默认启动服务中。重启系统后就不需要再次执行第一条命令了。

4.1.3 登录 xrdp

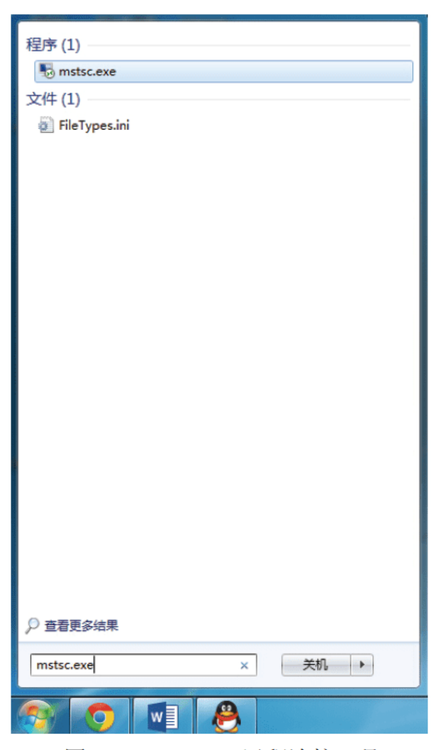
xrdp 服务启动后，可以在其他的远程主机上连接 Raspberry 的桌面。

1. Windows 登录

在 Windows 7 下远程桌面连接 Raspberry。

（1）单击“开始”菜单，输入 mstsc，运行 mstsc.exe。如图 4-1 所示。

（2）打开 Windows 远程桌面连接，输入 Raspberry 的 IP，如图 4-2 所示。



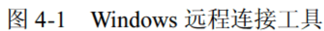


图 4-1 Windows 远程连接工具



图 4-2 Windows 远程桌面连接

（3）单击“连接”按钮，得到 Raspberry 的登录界面，如图 4-3 所示。

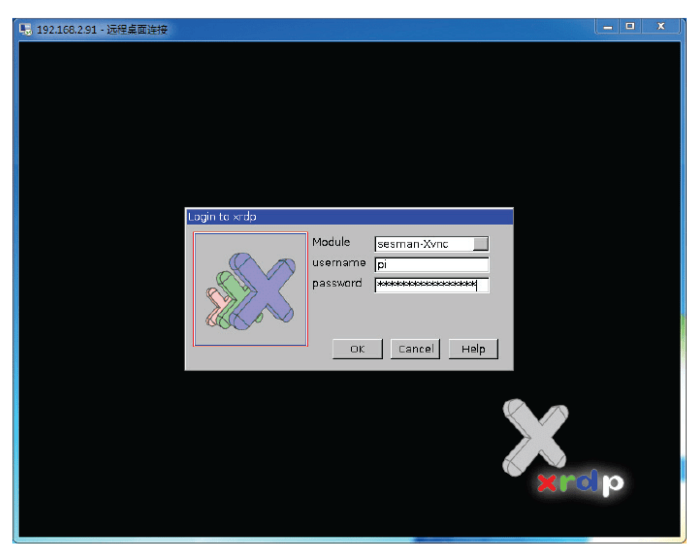


图 4-3　登录 Raspberry

（4）输入用户名，密码。单击 OK 按钮，得到了 Raspberry 的远程桌面，如图 4-4 所示。

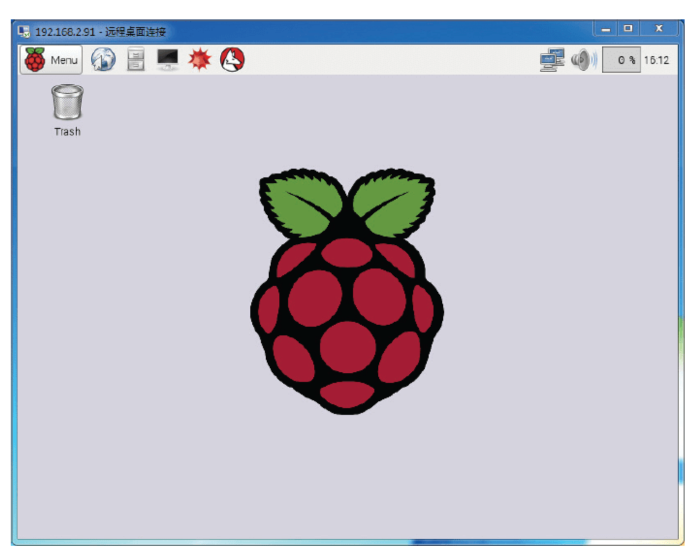


图 4-4　Windows 7 登录 Rasbperry 远程桌面

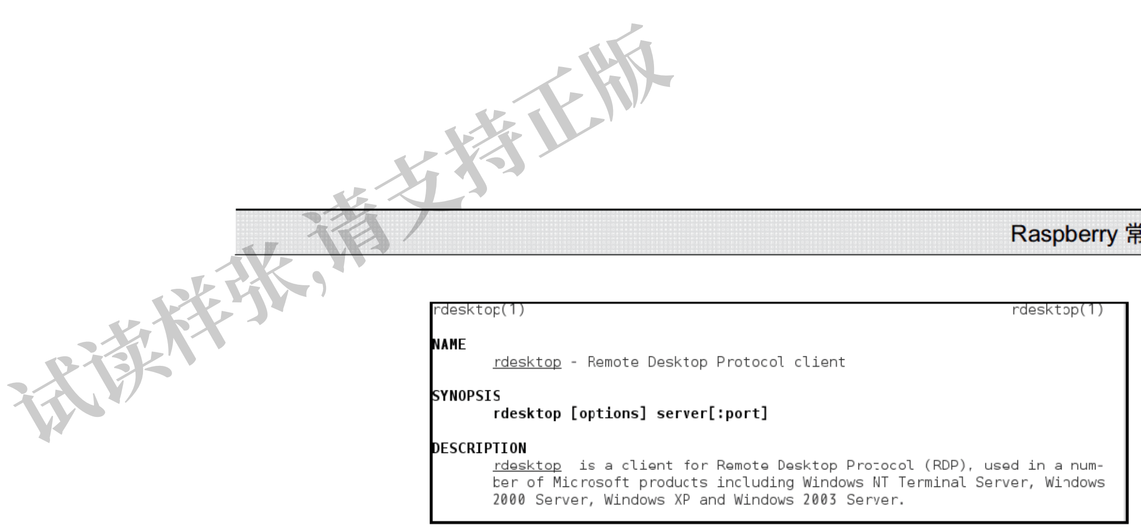
2. Linux 登录

在 Linux 下远程连接 Raspberry。Linux 下的远程桌面工具是 rdesktop，如果未安装 rdesktop，su 到 root 执行命令：

```
apt-get install rdesktop
```

（1）命令简介

先看下 man rdesktop，如图 4-5 所示：



```
rdesktop(1)                                                   rdesktop(1)

NAME
       rdesktop - Remote Desktop Protocol client

SYNOPSIS
       rdesktop [options] server[:port]

DESCRIPTION
       rdesktop  is a client for Remote Desktop Protocol (RDP), used in a num-
       ber of Microsoft products including Windows NT Terminal Server, Windows
       2000 Server, Windows XP and Windows 2003 Server.
```

图 4-5　man rdesktop

rdesktop 的常用参数如下：

- -f：全屏。
- -a：16 位色。
- -u：登录用户，可选。
- -p：登录密码，可选。
- -r：clipboard:PRIMARYCLIPBOARD 重要，剪贴板可以与远程桌面交互。
- -a：16 颜色，可选，不过最高就是 16 位。
- -z：压缩，可选。
- -g：1024 × 768 分辨率，可选，默认是一种比当前本地桌面低的分辨率。
- -P：缓冲，可选。
- -r：disk:wj=/home/magicgod 映射虚拟盘，可选，会在远程机器的网上邻居里虚拟出一个映射盘，功能很强，甚至可以是软盘或光盘。
- -r：sound:off 关闭声音，当然也可以把远程发的声音映射到本地来。

（2）实验测试

使用 rdesktop 远程连接 Raspberry 桌面，如图 4-6 所示，执行命令：

```
rdesktop -f 192.168.2.91 -u pi
```

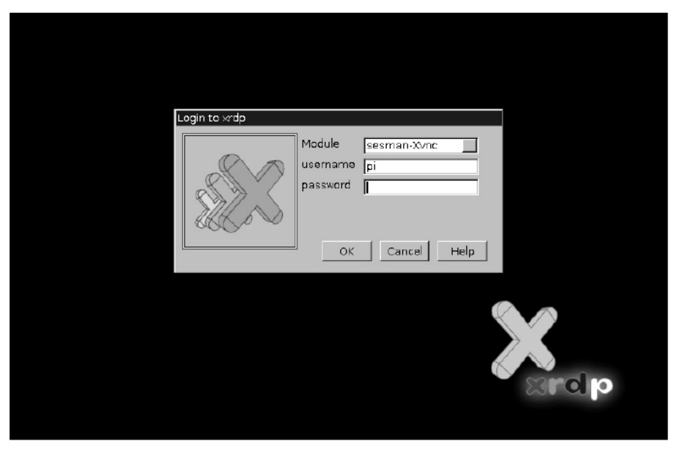


图 4-6　rdesktop 连接

输入密码后，单击 OK 按钮，如图 4-7 所示。

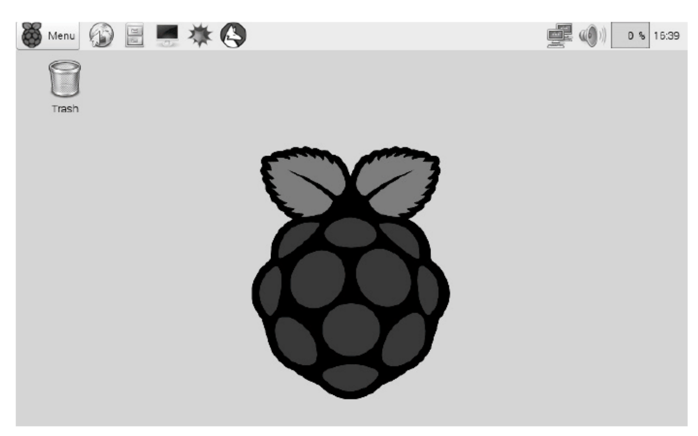


图 4-7　Linux 登录 Raspberry 远程桌面

利用 xrdp 服务连接 Raspberry 的优点是效果直观，缺点是速度较慢。Raspberry 所有的操作都是可以使用命令行的。一般使用 ssh 连接足矣，Xrdp 服务只作为补充，了解了解就可以了。



注　意

在 Linux 上，ssh 可以完成 99.99%的工作。即使想浏览网页，也有 lynx 浏览器、w3m 浏览器……可供挑选（无法浏览图片）。在熟悉 Linux 后，xrdp 功能的存在感会变得很低很低。

4.2　samba 共享服务

得益于目前硬件的价格越来越便宜，在同一局域网内可能有 Windows 主机、Linux 主机、Mac 主机甚至可能有 Unix 主机。在这么多不同的系统之间，怎么共享文件呢？首选方案 samba，万能的 samba。

4.2.1　samba 简介

samba 是一套使用 SMB（Server Message Block）协议的应用程序，通过支持这个协议，samba 允许 Linux 服务器与 Windows 系统之间进行通信，使跨平台的互访成为可能。samba 采用 C/S 模式，其工作机制是让 NetBIOS（Windows 网上邻居的通信协议）和 SMB 两个协议运行于 TCP/IP 通信协议之上，并且用 NetBEUI 协议让 Windows 在“网上邻居”中能浏览 Linux 服务器。

4.2.2　samba 安装

samba 可以说是 Linux 上最常用的一个服务了。得益于 Debian 良好的包管理系统 apt-get，在 RaspBian 上安装 samba 服务很简单。用 Putty 以默认用户 pi 登录到 Raspberry，执行命令：

```
sudo apt-get install samba
```

samba 安装完毕。

4.2.3　samba 配置

Raspberry 中 samba 服务的配置文件是/etc/samba/smb.conf，在配置 samba 服务之前先说明一下 samba 的安全等级。samba 有 4 种安全等级：

- security=share：用户访问 samba 服务器不需要提供用户名和口令，安全性能较低。
- security=user：samba 服务器默认的安全等级，每一个共享目录只能被一定的用户访问，并由 samba 服务器负责检查账号和密码的正确性。
- security=server：服务器安全级别，依靠其他 Windows NT/2000 或 samba 服务器来验证用户的账号和密码，是一种代理验证。此种安全模式下，系统管理员可以把所有的 Windows 用户和口令集中到一个 NT 系统上，使用 Windows NT 进行 samba 认证，远程服务器可以自动认证全部用户和口令，如果认证失败，samba 将使用用户级安全模式作为替代的方式。
- security=domain：域安全级别，使用主域控制器（PDC）来完成认证。

因为是个人使用，家庭用户直接选用 share 就可以了。如果有其他的用途，请自行参考配置。好了，先看下最简单的 smb.conf 是怎么设置的。顺便复习一下上章学习的 grep 命令，执行命令：

```
cd /etc/samba/
grep -v ^\; smb.conf | grep -v ^# | grep -v ^$
```

第二条命令的作用是将 smb.conf 中的注释行、空行去除，剩下的就是有用的设置了。结果如图 4-8 所示。

```
pi@raspberrypi: /etc/samba
pi@raspberrypi /etc/samba $ grep -v ^\; smb.conf | grep -v ^# | grep -v ^$
[global]
   workgroup = WORKGROUP
   server string = %h server
   dns proxy = no
   log file = /var/log/samba/log.%m
   max log size = 1000
   syslog = 0
   panic action = /usr/share/samba/panic-action %d
   encrypt passwords = true
   passdb backend = tdbsam
   obey pam restrictions = yes
   unix password sync = yes
   passwd program = /usr/bin/passwd %u
   passwd chat = *Enter\snew\s*\spassword:* %n\n *Retype\snew\s*\spassword:* %n\
n *password\supdated\ssuccessfully* .
   pam password change = yes
   map to guest = bad user
   usershare allow guests = yes
[homes]
   comment = Home Directories
   browseable = no
   read only = yes
   create mask = 0700
   directory mask = 0700
   valid users = %S
[printers]
   comment = All Printers
   browseable = no
   path = /var/spool/samba
   printable = yes
   guest ok = no
   read only = yes
   create mask = 0700
[print$]
   comment = Printer Drivers
   path = /var/lib/samba/printers
   browseable = yes
   read only = yes
   guest ok = no
pi@raspberrypi /etc/samba $
```

图 4-8　grep smb.conf

以上的设置有很多都不是必需的，而且少了最重要的 security 项。下面来配置一个最简单的 smb.conf 试试看。

备份 smb.conf 后，利用 smb.conf 的有效设置，稍微修改一下。执行命令：

```
cd /etc/samba
sudo cp smb.conf smb.conf.old
sudo sed -i -e '/^#/d' smb.conf  -e '/^\;/d' -e '/^$/d' -e '/homes/,$d'
```

使用 sed 去除了空行、注释行，并把 homes 以下的设置都去除了。在 global 中添加 security 选项，再仿照[homes]选项做一个[pi]的选项。以下是/etc/smb.conf 的代码：

```
 1 [global]
 2   workgroup = WORKGROUP
 3   server string = %h server
 4   dns proxy = no
 5   log file = /var/log/samba/log.%m
 6   max log size = 1000
 7   syslog = 0
 8   panic action = /usr/share/samba/panic-action %d
 9   encrypt passwords = true
10   passdb backend = tdbsam
11   obey pam restrictions = yes
12   unix password sync = yes
13   passwd program = /usr/bin/passwd %u
14   passwd chat = *Enter\snew\s*\spassword:* %n\n *Retype\snew\s*\spassword:*
%n\n *password\supdated\ssuccessfully* .
15   pam password change = yes
16   map to guest = bad user
17   usershare allow guests = yes
18
19
20
21 ####  用户自行添加
22 ##安全级别 share
23   security = share
24   guest account = pi
25   directory mask = 0700
26   create mask = 0700
27
28 [pi]
29 ## samba 注释
30   comment = User pi Home Directories
31 ## 共享目录
32   path = /home/pi
33   browseable = yes
34   writable = yes
35   public = yes
36   valid users = pi
37   read only = no
38   available = yes
39   guest ok = yes
```

这里有两个地方一定要注意，guest account 是一定要有的，guest ok 这个也是一定要有的。否则，即使设置的 security=share，在使用 Windows 登录时系统依然会提示要求输入密码，而且不管

输入什么密码都无法登录到 samba 服务器。

在 debian 中，有个 testparm 命令，可以测试 smb.conf 文件是否配置正确。可惜 RaspBian 没有这个命令，所以请再三检查 smb.conf 文件，避免犯一些低级错误。

配置完毕后，重启 samba 服务，并将 samba 服务添加到系统默认启动的服务中去。执行命令：

```
sudo /etc/init.d/samba start
sudo update-rc.d samba defaults
```

使用 smbclient 命令查看 smb 服务，执行命令：

```
smbclient -L //192.168.2.91
```

提示输入密码，无须输入，直接按 Enter 键，得到的结果如图 4-9 所示。

```
pi@raspberrypi: /etc/samba
pi@raspberrypi /etc/samba $ smbclient -L //192.168.2.91
WARNING: The security=share option is deprecated
Enter pi's password:
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.6.6]

        Sharename       Type      Comment
        ---------       ----      -------
        IPC$            IPC       IPC Service (raspberrypi server)
        pi              Disk      User pi Home Directories
Domain=[WORKGROUP] OS=[Unix] Server=[Samba 3.6.6]

        Server               Comment
        ---------            -------
        DEBIAN               debian server
        RASPBERRYPI          raspberrypi server

        Workgroup            Master
        ---------            -------
        WORKGROUP            DEBIAN
pi@raspberrypi /etc/samba $
```

图 4-9　smbclient –L //192.168.2.91

samba 服务配置完毕，基本上这样就可以使用了。美中不足的是磁盘比较小。如果已经挂载了大容量硬盘，将它加入 smb.conf 中共享起来，一个私人的文件服务器就建立好了。

4.2.4　登录 samba 服务器

1. Windows 环境登录 samba 服务器

samba 服务器已经设置完毕了，现在可以在 Windows 环境登录 samba 服务器了。在桌面上双击“网络”图标，如图 4-10 所示。



图 4-10　Windows 网络

显示在 workgroup 工作组里有三台主机，其中 RASPBERRYPI 就是刚才设置了 samba 服务器

的那台 Raspberry。双击 RASPBEYPI 的图标，如图 4-11 所示。

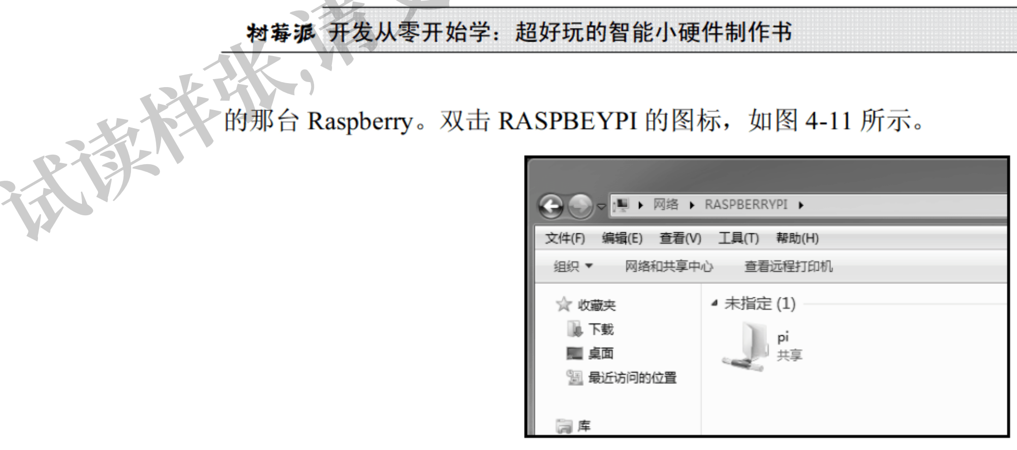


图 4-11 samba 共享目录

显示在 Raspberry 主机上的共享目录，也就是刚才在 smb.conf 里设置的共享文件夹 pi。双击 pi 图标，如图 4-12 所示。



图 4-12 samba 共享文件夹

使用 xrdp 服务，连接到 Raspberry 主机，看看 pi 用户的家目录，比较一下。如图 4-13 所示。

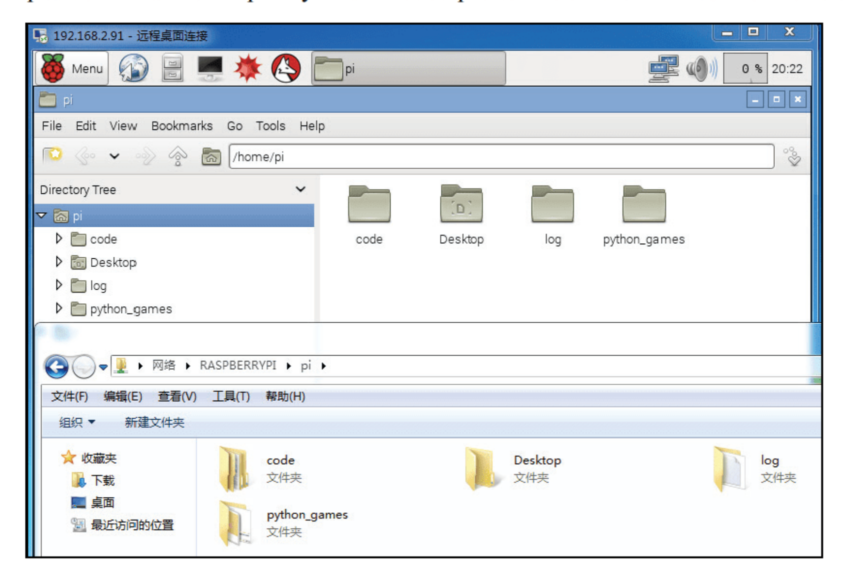


图 4-13 Raspberry samba

对比一下，完全一致。samba 配置成功。

2. Linux 环境登录 samba 服务器

Linux 的桌面环境很多，这里只测试 Gnome classic 环境。单击桌面左上方的“位置”图标，单击“连接到服务器”选项，如图 4-14 所示。



图 4-14 Linux Gnome

在“类型”下拉框中选择“Windows 共享”，在“服务器”文本框中填入 Raspberry 的 IP 地址，因为 samba 主机设置的安全级别是 share，所以这里除了 IP 外，其他的什么都不需要填。如图 4-15 所示。

图 4-15 Gnome 连接 samba

单击“连接”按钮，出现如图 4-16 所示界面：



图 4-16 Gnome 浏览共享目录

双击 pi 文件夹，出现如图 4-17 所示界面。



图 4-17 Gnome 浏览共享文件

好了，最后连接到 xrdp 服务器上比较一下，如图 4-18 所示。



图 4-18　Gnome samba

samba 服务器演示完毕。这里只演示了最简单的 samba 服务器配置，家庭、个人使用是没问题的。如有其他的用途，请自行调节安全级别。



注　意

使用 samba 服务器再加上 rsync 可以建立自己的私有云。这个小项目并不复杂。稍微运作一下就可以了。

4.3　miniDLNA 共享影音服务

samba 安装配置好后，在局域网内共享文件已经不是问题了。但在局域网播放影音文件却不是 samba 所擅长的。这里就要用到 miniDLNA 了。

4.3.1　miniDLNA 简介

DLNA 的全称是 DIGITAL LIVING NETWORK ALLIANCE（数字生活网络联盟），其宗旨是 Enjoy your music、photos and videos，anywhere anytime。DLNA 由索尼、英特尔、微软等发起成立，旨在解决个人 PC、消费电器、移动设备在内的无线网络和有线网络的互联互通，使得数字媒体和内容服务的无限制的共享和增长成为可能，目前成员公司已达 280 多家。

而 miniDLNA 按照 man minidlnad 的解释是 minidlnad——lightweight DLNA/UPnP-AV server。将 miniDLNA 安装好后，就可以在手机、平板等移动设备上播放 mniDLNA 服务器上的影音文件，无须再次下载。速度比使用 samba 的播放快很多。

4.3.2 miniDLNA 安装

使用 Putty 登录 Raspberry，执行命令：

```
sudo apt-get install minidlna
```

miniDLNA 安装完毕。

4.3.3 miniDLNA 配置

miniDLNA 的配置文件是/etc/minidlna.conf。这个文件配置起来很简单，先来看看默认的有效配置，如图 4-19 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ grep -v "#" /etc/minidlna.conf | grep -v ^$
media_dir=/var/lib/minidlna
port=8200
serial=12345678
model_number=1
album_art_names=Cover.jpg/cover.jpg/AlbumArtSmall.jpg/albumartsmall.jpg/AlbumArt
.jpg/albumart.jpg/Album.jpg/album.jpg/Folder.jpg/folder.jpg/Thumb.jpg/thumb.jpg
pi@raspberrypi ~ $
```

图 4-19 miniDLNA 默认配置

先将它备份一下，执行命令：

```
sudo cp /etc/minidlna.conf /etc/minidlna.conf.bak
```

创建 dlna 的家目录：

```
sudo mkdir -pv /mnt/disk/dlna/audio
sudo mkdir -pv /mnt/disk/dlna/video
```

将配置文件稍微修改一下，以下是修改好的/etc/minidlan.conf。

```
####  音频文件夹
media_dir=A,/mnt/disk/dlna/audio
####  视频文件夹
media_dir=V,/mnt/disk/dlna/video
port=8200
serial=12345678
model_number=1
album_art_names=Cover.jpg/cover.jpg/AlbumArtSmall.jpg/albumartsmall.jpg/AlbumA
rt.jpg/albumart.jpg/Album.jpg/album.jpg/Folder.jpg/folder.jpg/Thumb.jpg/thumb.
jpg
```

然后重启 minidlna 服务，将它添加到启动服务中去，执行命令：

```
sudo /etc/init.d/minidlna restart
sudo update-rc.d minidlna defaults
```

好了，miniDLNA 服务器配置完毕了。以后就可以用支持 dlna 的播放器直接播放服务器上的影音文件了。



注 意

目前 DLNA 默认支持的图像格式是 jpg、png、gif、tiff。音频格式是 mp3、aac、ac-3、atrac-3+、wma9。视频格式是 mpeg-1、mpeg-2、mpeg-4、avc、wmv9。没有流行的 rmvb，也没有高清的 mkv。如果想添加新的格式，得自行 DIY。看来革命尚未成功，同志仍需努力啊。另外就是在 iOS 上支持 dlna 的播放器不多，即使有也是收费版的。在这点 Android 平台做得不错。

4.4 VSFTP FTP 服务

在局域网内共享文件，除了 samba 外还有 FTP 服务器。在 Linux 下首屈一指的 FTP 服务器就要数 VSFTP 了。它是一个老牌的明星软件，时间证明了实力。

4.4.1 VSFTP 简介

VSFTP 是一个基于 GPL 发布的类 Unix 系统上使用的 FTP 服务器软件，它的全称是 Very Secure FTP。从此名称可以看出来，编制者的初衷是代码的安全。安全性是编写 VSFTP 的初衷，除了这与生俱来的安全特性以外，高速与高稳定性也是 VSFTP 的两个重要特点。

在速度方面，使用 ASCII 代码的模式下载数据时，VSFTP 的速度是 Wu-FTP 的两倍，如果 Linux 主机使用 2.4.*的内核，在千兆以太网的下载速度可达 86MB/S。

在稳定方面，VSFTP 就更加地出色，VSFTP 在单机（非集群）上支持 4000 个以上的并发用户同时连接，根据 Red Hat 的 FTP 服务器的数据，VSFTP 服务器可以支持 15000 个并发用户。

仅作为个人服务器，VSFTP 是完全合格的。

4.4.2 VSFTP 安装

VSFTP 很小，但功能强大。在配置好 apt-get 源后，安装其他软件就很方便了。这里安装 VSFTP 只需要一条命令就可以了。执行命令：

```
sudo apt-get install vsftpd
```

4.4.3 vsftp 配置

vsfptd 服务的配置文件是/etc/vsftpd.conf。为避免破坏文件，还是将该配置文件做个备份。执行命令：

```
sudo cp /etc/vsftpd.conf /etc/vsftpd.conf.bak
sudo vi /etc/vsftpd.conf
```

下面是修改后的 vsftpd.conf。

```
1 # Example config file /etc/vsftpd.conf
2 #
```

```
 3 # The default compiled in settings are fairly paranoid. This sample file
 4 # loosens things up a bit, to make the ftp daemon more usable.
 5 # Please see vsftpd.conf.5 for all compiled in defaults.
 6 #
 7 # READ THIS: This example file is NOT an exhaustive list of vsftpd options.
 8 # Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
 9 # capabilities.
10 #
11 #
12 # Run standalone?  vsftpd can run either from an inetd or as a standalone
13 # daemon started from an initscript.
14 listen=YES
15 #
16 # Run standalone with IPv6?
17 # Like the listen parameter, except vsftpd will listen on an IPv6 socket
18 # instead of an IPv4 one. This parameter and the listen parameter are mutually
19 # exclusive.
20 #listen ipv6=YES
21 #
22
23
24 # Allow anonymous FTP? （Beware - allowed by default if you comment this out）.
25 #*-------  是否允许匿名用户登录
26 #anonymous_enable=YES
27 anonymous_enable=NO
28
29
30 # Uncomment this to allow local users to log in.
31 #*-------  是否允许本地用户登录，在这里本地用户只有pi
32 #local_enable=NO
33 local_enable=YES
34
35
36 # Uncomment this to enable any form of FTP write command.
37 #*------  是否允许本地用户对FPT服务器有写入权限
38 #write enable=NO
39 write_enable=NO
40
41
42
43 # Default umask for local users is 077. You may wish to change this to 022,
44 # if your users expect that （022 is used by most other ftpd's）
45 #*------  本地用户的文件掩码
46 #local_umask=022
47 local_umask=022
48
49
50
51 # Uncomment this to allow the anonymous FTP user to upload files. This only
```

```
52 # has an effect if the above global write enable is activated. Also, you will
53 # obviously need to create a directory writable by the FTP user.
54 #*------  是否允许匿名用户上传文件
55 #anon_upload_enable=YES
56 anon_upload_enable=NO
57
58
59 #
60 # Uncomment this if you want the anonymous FTP user to be able to create
61 # new directories.
62 #*------  是否允许匿名用户创建新文件夹
63 #anon_mkdir_write_enable=YES
64 anon_mkdir_write_enable=NO
65
66
67 # Activate directory messages - messages given to remote users when they
68 # go into a certain directory.
69 #*------  是否显示欢迎信息
70 dirmessage_enable=YES
71
72
73 #
74 # If enabled, vsftpd will display directory listings with the time
75 # in  your  local  time  zone.  The default is to display GMT. The
76 # times returned by the MDTM FTP command are also affected by this
77 # option.
78 use_localtime=YES
79 #
80 # Activate logging of uploads/downloads.
81 xferlog_enable=YES
82 #
83 # Make sure PORT transfer connections originate from port 20 (ftp-data).
84 #*------  设定 ftp 服务数据端口
85 connect_from_port_20=YES
86
87
88 #
89 # If you want, you can arrange for uploaded anonymous files to be owned by
90 # a different user. Note! Using "root" for uploaded files is not
91 # recommended!
92 #*------  是否允许修改上传文件的属主
93 #chown_uploads=NO
94 chown_uploads=YES
95 #*------  如果允许，输入该属主的用户名
96 #chown_username=whoever
97 chown_username=pi
98
99
100 #
```

```
101 # You may override where the log file goes if you like. The default is shown
102 # below.
103 #*------  设置日志文件
104 #xferlog_file=/var/log/vsftpd.log
105 xferlog_file=/home/pi/log/vsftpd.log
106
107
108 #
109 # If you want, you can have your log file in standard ftpd xferlog format.
110 # Note that the default log file location is /var/log/xferlog in this case.
111 #*------  设置 log 文件格式为标准 xferlog
112 #xferlog_std_format=YES
113 xferlog_std_format=YES
114
115
116 #
117 # You may change the default value for timing out an idle session.
118 #*------  设置数据传输中断间隔时间
119 #idle_session_timeout=600
120 idle_session_timeout=600
121
122
123 #
124 # You may change the default value for timing out a data connection.
125 #*------ 设置数据连接超时时间
126 #data_connection_timeout=120
127 data_connection_timeout=120
128
129
130 #
131 # It is recommended that you define on your system a unique user which the
132 # ftp server can use as a totally isolated and unprivileged user.
133 #nopriv_user=ftpsecure
134 #
135 # Enable this and the server will recognise asynchronous ABOR requests. Not
136 # recommended for security （the code is non-trivial）. Not enabling it,
137 # however, may confuse older FTP clients.
138 #async_abor_enable=YES
139 #
140 # By default the server will pretend to allow ASCII mode but in fact ignore
141 # the request. Turn on the below options to have the server actually do ASCII
142 # mangling on files when in ASCII mode.
143 # Beware that on some FTP servers, ASCII support allows a denial of service
144 # attack （DoS） via the command "SIZE /big/file" in ASCII mode. vsftpd
145 # predicted this attack and has always been safe, reporting the size of the
146 # raw file.
147
148
149 # ASCII mangling is a horrible feature of the protocol.
```

```
150 #*------  选择用 ASCII 方式上传下载
151 #ascii_upload_enable=YES
152 #ascii_download_enable=YES
153 ascii_upload_enable=YES
154 ascii_download_enable=YES
155
156
157 #
158 # You may fully customise the login banner string:
159 #ftpd_banner=Welcome to blah FTP service.
160 #
161 # You may specify a file of disallowed anonymous e-mail addresses. Apparently
162 # useful for combatting certain DoS attacks.
163 #deny_email_enable=YES
164 # (default follows)
165 #banned_email_file=/etc/vsftpd.banned_emails
166 #
167 # You may restrict local users to their home directories.  See the FAQ for
168 # the possible risks in this before using chroot_local_user or
169 # chroot_list_enable below.
170 #chroot_local_user=YES
171 #
172
173
174 # You may specify an explicit list of local users to chroot() to their home
175 # directory. If chroot_local_user is YES, then this list becomes a list of
176 # users to NOT chroot().
177 # (Warning! chroot'ing can be very dangerous. If using chroot, make sure that
178 # the user does not have write access to the top level directory within the
179 # chroot)
180 #chroot_local_user=YES
181 #chroot_list_enable=YES
182 # (default follows)
183 chroot_list_file=/etc/vsftpd.chroot_list
184 #
185 # You may activate the "-R" option to the builtin ls. This is disabled by
186 # default to avoid remote users being able to cause excessive I/O on large
187 # sites. However, some broken FTP clients such as "ncftp" and "mirror" assume
188 # the presence of the "-R" option, so there is a strong case for enabling it.
189 #ls_recurse_enable=YES
190 #
191 # Customization
192 #
193 # Some of vsftpd's settings don't fit the filesystem layout by
194 # default.
195 #
196 # This option should be the name of a directory which is empty.  Also, the
197 # directory should not be writable by the ftp user. This directory is used
198 # as a secure chroot() jail at times vsftpd does not require filesystem
```

```
199 # access.
200 secure_chroot_dir=/var/run/vsftpd/empty
201 #
202 # This string is the name of the PAM service vsftpd will use.
203 pam_service_name=vsftpd
204 #
205 # This option specifies the location of the RSA certificate to use for SSL
206 # encrypted connections.
207 rsa_cert_file=/etc/ssl/private/vsftpd.pem
208
209
210 #*------  设置本地用户登录目录
211 local_root=/home/pi/ftp
```

好了，配置文件修改完毕，如果去除掉空行和注释行，这个配置并不多。创建 ftp 服务的家目录，执行命令：

```
mkdir -pv ~/ftp
sudo /etc/init.d/vsftpd start
sudo update-rc.d samba defaults
```

将 vsftpd 设置成开机启动服务。

4.4.4 登录 VSFTP 服务器

1. Windows 环境下登录 VSFTP 服务器

Windows 环境下登录 FTP 的软件很多，这里选择以 secureFX 为例，如图 4-20 所示。

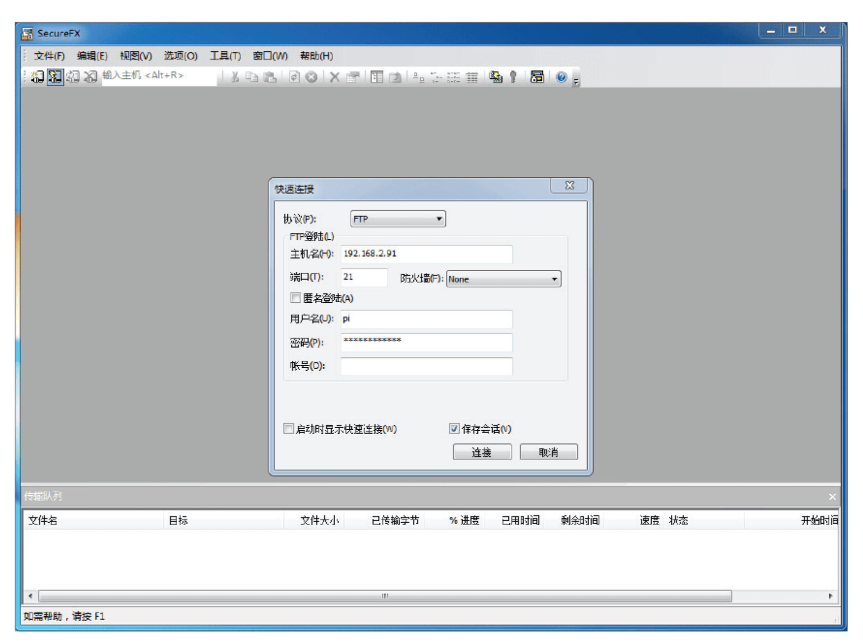


图 4-20 secureFX

协议选择 FTP 协议，主机名文本框填入 Raspberry 的 IP 地址（192.168.2.91），用户名文本框中填入 Raspberry 默认的用户名，最后在密码文本框中填入 pi 用户的密码。单击“连接”按钮后，如图 4-21 所示。

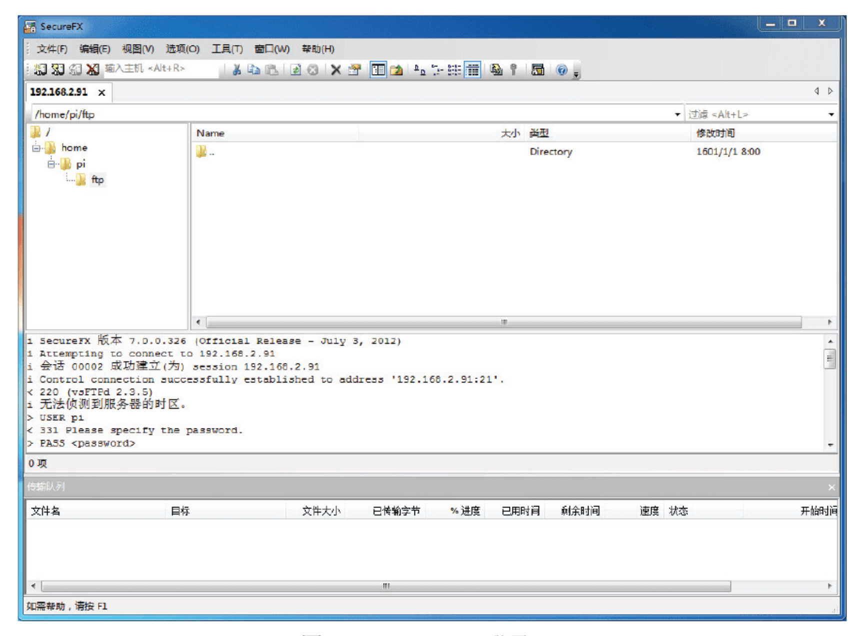


图 4-21　secureFx 登录 ftp

2. Linux 环境下登录 VSFTP 服务器

在 Linux 下也有 FTP 的图像化客户端，例如 filezilla 之类的软件。它们登录 FTP 服务器的方法与 Windows 很相似。这里只演示用 FTP 命令登录 FTP 服务器。

（1）参数简介

首先还是来看 man ftp，如图 4-22 所示。

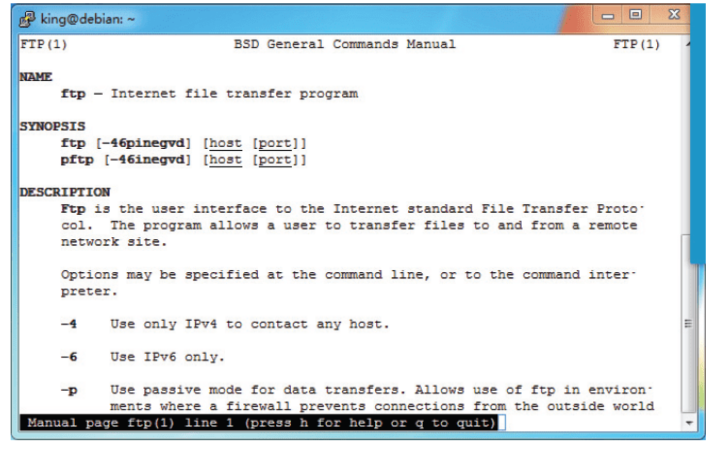


图 4-22　man ftp

ftp 常用参数：

- -4: use IPv4 addresses only
- -6: use IPv6, nothing else
- -p: enable passive mode (default for pftp)
- -i: turn off prompting during mget
- -n: inhibit auto-login
- -e: disable readline support, if present
- -g: disable filename globbing
- -v: verbose mode
- -t: enable packet tracing [nonfunctional]
- -d: enable debugging

（2）实战测试

使用 Putty 登录 Linux(Debian)环境或者直接登录 Linux。使用 FTP 命令登录服务端。如图 4-23 所示。



图 4-23　ftp 命令登录服务端

操作完毕后。使用 bye 命令退出。

使用 FTP 命令登录到服务端，使用命令来上传、下载、创建、删除文件。这里就不多做介绍了，有兴趣的可以自行谷歌百度。

本来还应该将登录用户设置成无法 chroot 到其他目录的。但 Raspberry 上的 vsftpd 略有瑕疵，再加上个人用户无须如此麻烦，暂时就只能这样了。如果想追求完美，请自行下载 VSFTP，编译安装后配置。


注　意

存在即为合理，FTP 服务器能够经久不衰，时间已证明了它的强大。

4.5　Nginx

在网络上使用率最高的服务还是 www 服务，而 www 服务最出名的两个明星就要数 Nginx 和 Apache 了。先来看 Nginx 吧。

4.5.1 Nginx 简介

Nginx 是一款轻量级的 Web 服务器/反向代理服务器及电子邮件（IMAP/POP3）代理服务器，并在一个 BSD-like 协议下发行。由俄罗斯的程序设计师 Igor Sysoev 所开发，供俄国大型的入口网站及搜索引擎 Rambler（俄文：Рамблер）使用。其特点是占有内存少，并发能力强，事实上 Nginx 的并发能力确实在同类型的网页服务器中表现较好。如果建站只要求静态网页，还是建议使用 Nginx。相比 Apache 而言，它更加小巧，性能稳定，非常适合 Raspberry 使用。

4.5.2 Nginx 安装

安装 Nginx 一条命令足矣，执行命令：

```
sudo apt-get install nginx
```

就这样 Nginx 已经安装完毕了。

4.5.3 Nginx 配置

Nginx 的配置文件是/etc/nginx/nginx.conf，在 http 服务上基本是不需要配置什么的，可以直接使用。但对于自建的 www 站点倒是可以配置一下。www 站点的配置文件是/etc/nginx/sites-available/default.bak。首先还是先备份该配置文件，执行命令：

```
cd /etc/nginx/sites-available
sudo cp default default.bak
mkdir ~/www
```

以下是 default 的源文件：

```
 1 # You may add here your
 2 # server {
 3 #   ...
 4 # }
 5 # statements for each of your virtual hosts to this file
 6
 7 ##
 8 # You should look at the following URL's in order to grasp a solid understanding
 9 # of Nginx configuration files in order to fully unleash the power of Nginx.
10 # http://wiki.nginx.org/Pitfalls
11 # http://wiki.nginx.org/QuickStart
12 # http://wiki.nginx.org/Configuration
13 #
14 # Generally, you will want to move this file somewhere, and start with a clean
15 # file but keep this around for reference. Or just disable in sites-enabled.
16 #
17 # Please see /usr/share/doc/nginx-doc/examples/ for more detailed examples.
18 ##
19
20 server {
```

```
21      #listen   80; ## listen for ipv4; this line is default and implied
22      #listen   [::]:80 default_server ipv6only=on; ## listen for ipv6
23
24      root /usr/share/nginx/www;
25      index index.html index.htm;
26
27      # Make site accessible from http://localhost/
28      server_name localhost;
29
30      location / {
31          # First attempt to serve request as file, then
32          # as directory, then fall back to displaying a 404.
33          try_files $uri $uri/ /index.html;
34          # Uncomment to enable naxsi on this location
35          # include /etc/nginx/naxsi.rules
36      }
37
38      location /doc/ {
39          alias /usr/share/doc/;
40          autoindex on;
41          allow 127.0.0.1;
42          allow ::1;
43          deny all;
44      }
45
46      # Only for nginx-naxsi used with nginx-naxsi-ui : process denied requests
47      #location /RequestDenied {
48      #   proxy_pass http://127.0.0.1:8080;
49      #}
50
51      #error_page 404 /404.html;
52
53      # redirect server error pages to the static page /50x.html
54      #
55      #error_page 500 502 503 504 /50x.html;
56      #location = /50x.html {
57      #   root /usr/share/nginx/www;
58      #}
59
60      # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
61      #
62      #location ~ \.php$ {
61      #
62      #location ~ \.php$ {
63      #   fastcgi_split_path_info ^(.+\.php)(/.+)$;
64      #   # NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
65      #
66      #   # With php5-cgi alone:
67      #   fastcgi_pass 127.0.0.1:9000;
```

```
68     #   # With php5-fpm:
69     #   fastcgi_pass unix:/var/run/php5-fpm.sock;
70     #   fastcgi_index index.php;
71     #   include fastcgi_params;
72     #}
73
74     # deny access to .htaccess files, if Apache's document root
75     # concurs with nginx's one
76     #
77     #location ~ /\.ht {
78     #   deny all;
79     #}
80 }
81
82
83 # another virtual host using mix of IP-, name-, and port-based configuration
84 #
85 #server {
86 #   listen 8000;
87 #   listen somename:8080;
88 #   server_name somename alias another.alias;
89 #   root html;
90 #   index index.html index.htm;
91 #
92 #   location / {
93 #       try_files $uri $uri/ =404;
94 #   }
95 #}
96
97
98 # HTTPS server
99 #
100 #server {
101 #   listen 443;
102 #   server_name localhost;
103 #
104 #   root html;
105 #   index index.html index.htm;
106 #
107 #   ssl on;
108 #   ssl_certificate cert.pem;
109 #   ssl_certificate_key cert.key;
110 #
111 #   ssl_session_timeout 5m;
112 #
113 #   ssl_protocols SSLv3 TLSv1;
114 #   ssl_ciphers ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv3:+EXP;
115 #   ssl_prefer_server_ciphers on;
116 #
```

```
117 #   location / {
118 #       try_files $uri $uri/ =404;
119 #   }
120 #}
```

这么长的配置文件，实际上需要修改的只有 21、24 两行。将其修改为：

```
 21     listen   80; ## listen for ipv4; this line is default and implied
 22
 23
 24     root /home/pi/www;
```

这样就可以了。然后在 www 站点的根目录，也就是/home/pi/www 下建立一个 index.html 文件。或者将/usr/share/nginx/www/index.html 复制过来，稍微修改一下，与原文件稍有区别即可。执行命令：

```
cp /usr/share/nginx/www/index.html /home/pi/www/
sed -I s
sed -i 's/nginx!/& I am raspberry pi!/g' index.html
sudo /etc/init.d/nginx
sudo /etc/init.d/nginx start
```

好了，现在打开浏览器，输入 Raspberry 的 ip 192.168.2.91，与 index.html 比较一下，如图 4-24 所示。

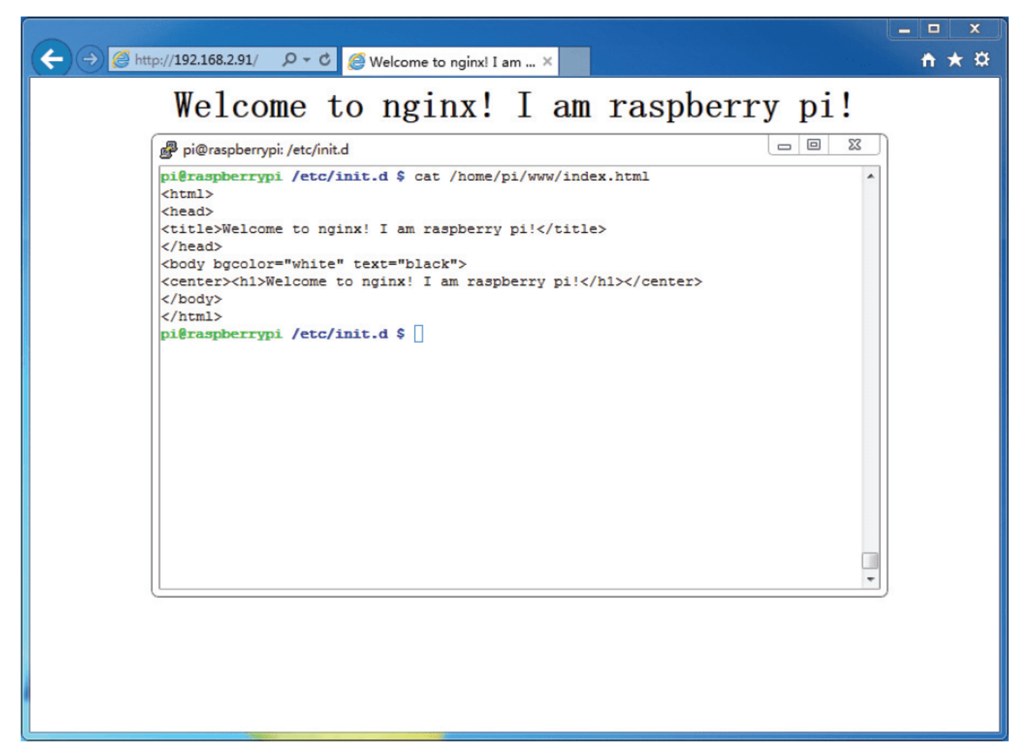


图 4-24　Nginx 服务器

好了，比较结果没有问题。Nginx 对静态网页的效果是非常不错的。如果只是需要一个 html 的网页，无须 ASP、PHP、数据库的网站，选择 Nginx 非常合适。如果是建立动态网页的网站，那还是选择 Apache 吧。



注 意

除了大名鼎鼎的 LAMP 外，Nginx 也有相应的套件 LNMP。它们只是在侧重点有所不同，在功能上 LNMP 毫不逊色。

4.6 LAMP

Nginx 介绍完毕，下面该说道 Apache 了。在 Linux 下 Apache 往往不会单独出现，和它一起出现的是一个著名的套装 LAMP。

4.6.1 LAMP 简介

LAMP 是 Linux+Apache+MySQL+PHP 的简称，是一组常用来搭建动态网站或者服务器的开源软件，本身都是各自独立的程序，但是因为常被放在一起使用，拥有了越来越高的兼容度，共同组成了一个强大的 Web 应用程序平台。在上一节曾说过，静态网页最好的选择是 Nginx。但动态网页还是使用 LAMP，它不仅配置简单方便，而且功能强大，资料众多，非常合适建设动态网站。

4.6.2 LAMP 安装

安装 LAMP 实际上就是分别把 Linux、Apache、MySQL、PHP 安装起来。再配合相应的联合软件，将它们组合到一起。执行命令：

```
sudo apt-get install apache2
sudo apt-get install php5
sudo apt-get install mysql-server
```

只需要安装主程序，其他依赖程序会自动安装。在安装 mysql-server 时，会要求设定 MySQL 的 root 用户密码，如图 4-25 所示。

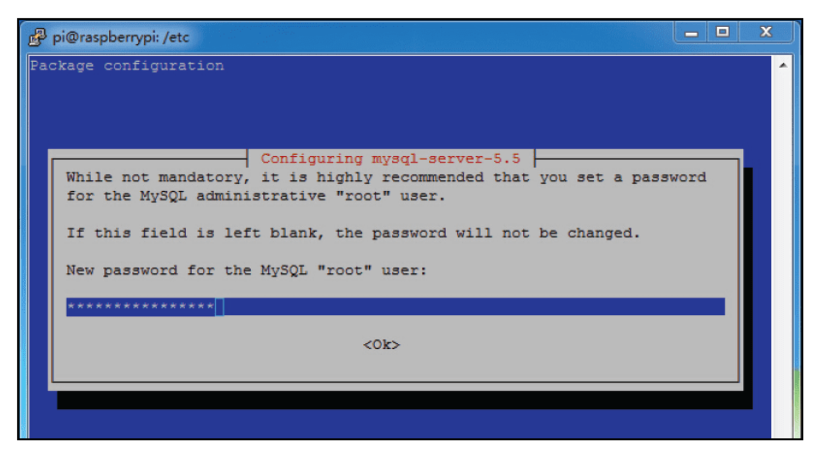


图 4-25 设置 MySQL 密码

再次输入 MySQL 的 root 用户密码，按 Enter 键，如图 4-26 所示。

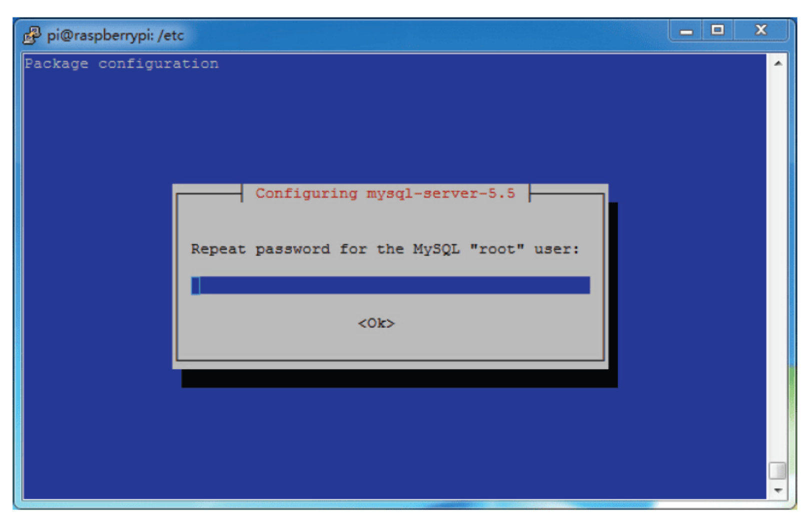


图 4-26　确认 MySQL 密码

好了，稍等片刻，MySQL 就安装完毕了。

安装软件将它们粘合起来。执行命令：

```
sudo apt-get install php5-mysql
```

到了这里 LAMP 就已经安装完毕了。下面开始配置 LAMP。

4.6.3　LAMP 配置

LAMP 中，可以配置的只有 Apache2，MySQL 和 PHP 都不需要配置了。首先要配置的是 Web 站点的端口。众所周知 Web 服务的端口是 80，但有时候 80 端口被占用或者不希望使用 80 端口时，那就只有修改 Apache2 的默认端口了。Apache2 端口配置文件是/etc/Apache2/ports.conf。执行命令：

```
cd /etc/Apache2
ls -l ports.conf
cp ports.conf ports.conf.bak
grep -v '#' ports.conf | grep -v ^$
```

得到有效的配置，如图 4-27 所示。

```
pi@raspberrypi: /etc/apache2
pi@raspberrypi /etc/apache2 $ grep -v '#' ports.conf |grep -v ^$
NameVirtualHost *:80
Listen 80
<IfModule mod_ssl.c>
    Listen 443
</IfModule>
<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
pi@raspberrypi /etc/apache2 $
```

图 4-27　Apache 端口设置

修改端口，只需要修改第 1 行和第 2 行就可以了。例如，想把 Web 服务端口修改成 8080，执行命令：

```
sudo sed -i 's/80/8080/g' ports.conf
```

最终的 ports.conf 代码如下:

```
 1 # If you just change the port or add more ports here, you will likely also
 2 # have to change the VirtualHost statement in
 3 # /etc/Apache2/sites-enabled/000-default
 4 # This is also true if you have upgraded from before 2.2.9-3 (i.e. from
 5 # Debian etch). See /usr/share/doc/Apache2.2-common/NEWS.Debian.gz and
 6 # README.Debian.gz
 7
 8 NameVirtualHost *:8080
 9 ####  Listen 是监听的端口，默认的 http 端口是 80
10 Listen 8080
11 <IfModule mod_ssl.c>
12     # If you add NameVirtualHost *:443 here, you will also have to change
13     # the VirtualHost statement in /etc/Apache2/sites-available/default-ssl
14     # to <VirtualHost *:443>
15     # Server Name Indication for SSL named virtual hosts is currently not
16     # supported by MSIE on Windows XP.
17     Listen 443
18 </IfModule>
19
20 <IfModule mod_gnutls.c>
21     Listen 443
22 </IfModule>
23
```

下一个要修改的是 Web 站点的主目录，默认情况下 Web 站点的主目录是/var/www。这样的缺点是/var/www 目录必须有特殊的权限才能修改、添加、删除文件，非常不方便。一般 Raspberry 都是个人使用单用户登录，可以将 Web 站点的主目录移动到 pi 用户的主目录下。如果觉得 Raspberry 空间小，也可以挂载一块大容量的移动硬盘到/mnt 下，然后将 Web 站点的主目录移动到新磁盘里。查看 Apache2 站点的配置文件/etc/Apache2/sites-available/default 的有效配置，如图 4-28 所示，执行命令：

```
cd /etc/Apache2/sites-available
sudo cp default default.bak
grep -v '#' default
```

```
pi@raspberrypi /etc/apache2/sites-available $ grep -v '#' default
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
pi@raspberrypi /etc/apache2/sites-available $ 
```

图 4-28　Apache 有效配置

只需要将‘/var/www’替换成‘/home/pi/www’，将 80 替换成 8080 就可以了。执行命令：

```
sudo sed -i 's/\/var\/www/\/home\/pi\/www/g' default
sudo sed -i 's/80/8080/g' default
```

修改完毕后，/etc/Apache2/sites-available/default 的代码如下：

```
 1 <VirtualHost *:8080>
 2    ServerAdmin webmaster@localhost
 3
 4    DocumentRoot /home/pi/www
 5    <Directory />
 6       Options FollowSymLinks
 7       AllowOverride None
 8    </Directory>
 9    <Directory /home/pi/www/>
10       Options Indexes FollowSymLinks MultiViews
11       AllowOverride None
12       Order allow,deny
13       allow from all
14    </Directory>
15
16    ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
17    <Directory "/usr/lib/cgi-bin">
18       AllowOverride None
19       Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
20       Order allow,deny
21       Allow from all
22    </Directory>
23
```

```
24     ErrorLog ${APACHE_LOG_DIR}/error.log
25
26     # Possible values include: debug, info, notice, warn, error, crit,
27     # alert, emerg.
28     LogLevel warn
29
30     CustomLog ${APACHE_LOG_DIR}/access.log combined
31  </VirtualHost>
```

在上一节中，曾将/home/pi/www 作为 Nginx 的主目录，这里就不修改了。直接借用 Nginx 的 index.html。所有配置完毕，下面启动 Apache2 服务、MySQL 服务。执行命令：

```
sudo /etc/init.d/Apache2 start
sudo /etc/init.d/MySQL start
```

现在打开浏览器，输入 Raspberry 的 ip 192.168.2.91:8080，与 index.html 比较一下，如图 4-29 所示。

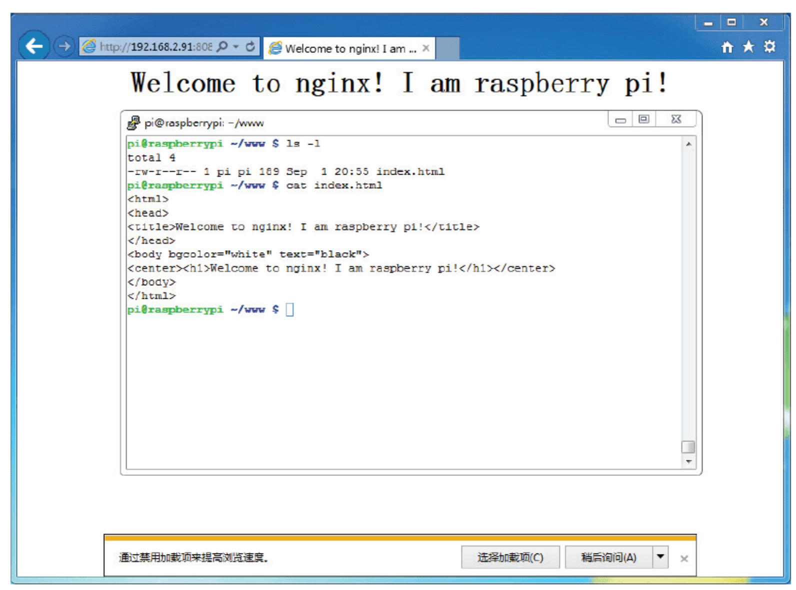
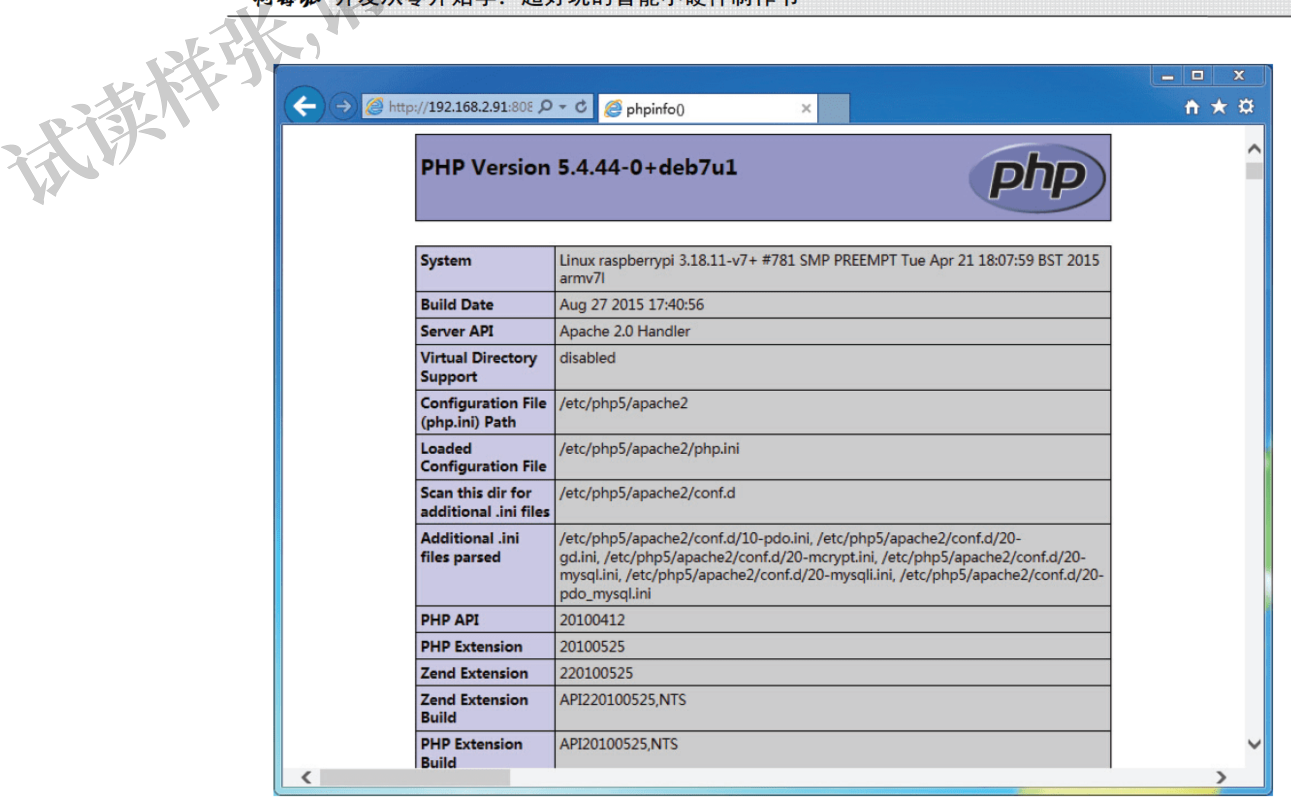


图 4-29　Apache2 服务器

好了，比较结果没有问题。最后来测试一下 PHP，执行命令：

```
cd /home/pi/www
echo "<?PHP PHPinfo() ; ?>" > PHPinfo.PHP
```

打开浏览器，打开 http://192.168.2.91:8080/PHPinfo.PHP。如图 4-30 所示。

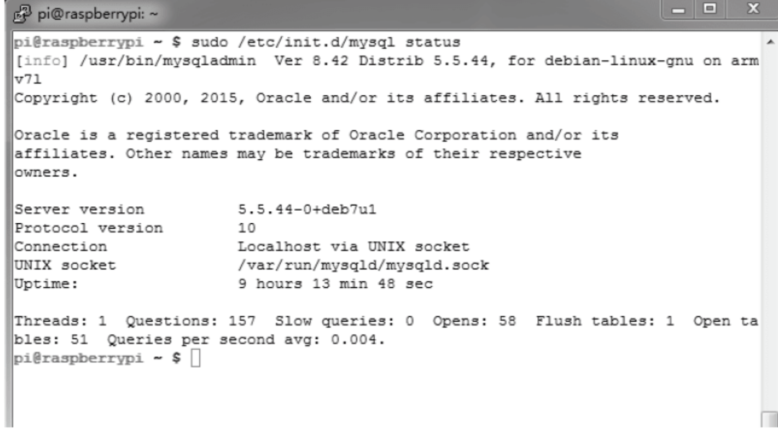


PHP Version 5.4.44-0+deb7u1

System	Linux raspberrypi 3.18.11-v7+ #781 SMP PREEMPT Tue Apr 21 18:07:59 BST 2015 armv7l
Build Date	Aug 27 2015 17:40:56
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-gd.ini, /etc/php5/apache2/conf.d/20-mcrypt.ini, /etc/php5/apache2/conf.d/20-mysql.ini, /etc/php5/apache2/conf.d/20-mysqli.ini, /etc/php5/apache2/conf.d/20-pdo_mysql.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS

图 4-30　PHPinfo.PHP

好了，PHP 测试通过。再来测试 MySQL，先看 MySQL 服务是否启动，如图 4-31 所示。



```
pi@raspberrypi ~ $ sudo /etc/init.d/mysql status
[info] /usr/bin/mysqladmin  Ver 8.42 Distrib 5.5.44, for debian-linux-gnu on arm
v7l
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Server version          5.5.44-0+deb7u1
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 9 hours 13 min 48 sec

Threads: 1  Questions: 157  Slow queries: 0  Opens: 58  Flush tables: 1  Open ta
bles: 51  Queries per second avg: 0.004.
pi@raspberrypi ~ $
```

图 4-31　查看服务

再检查 MySQL 服务的端口是否正常，如图 4-32 所示。

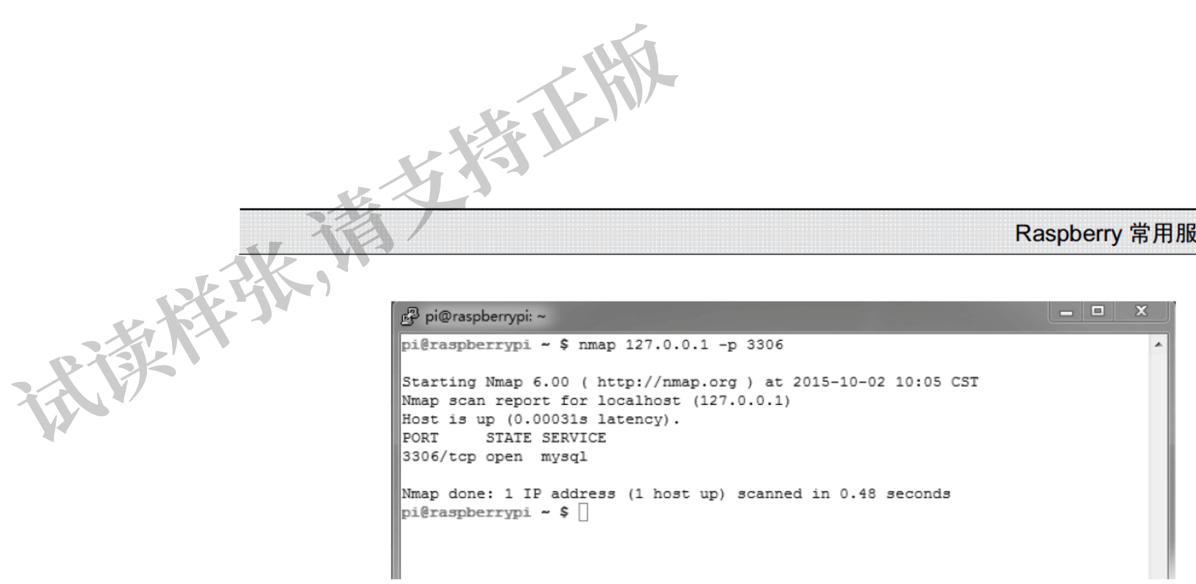


图 4-32 扫描端口

最后登录 MySQL，测试用户名和密码是否正确，如图 4-33 所示。

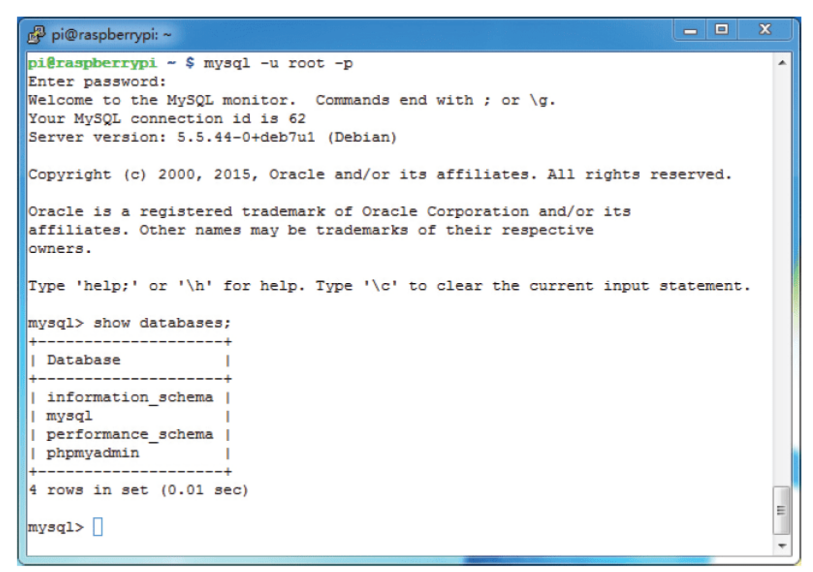


图 4-33 登录 MySQL

MySQL 测试完毕。最后，如果需要开机启动 Apache2 和 MySQL，执行命令：

```
sudo update-rc.d apache2 defaults
sudo update-rc.d mysql defaults
```

这样以后就不需要每次重启 Raspberry 后再输入命令启动服务了。有了这个完善的 LAMP 平台，再也不同担心没有试验平台学习 PHP 了。



注 意

LAMP 是目前最流行的 WWW 建站套件。最流行的不一定是最好的，但它一定是最方便，适应性最广泛的。

第 5 章 Raspberry常用功能

如果仅仅把 Raspberry 当成一个微型服务器用未免有些浪费了。配以合适的软件，它能发挥更大的作用。本章主要介绍 Raspberry 一些有趣的使用方法。从软件方面充分挖掘 Raspberry 的潜能。

本章主要内容包括：

- 如何挂载磁盘
- 如何利用 Aria2 下载机下载
- 如何使用迅雷远程下载
- 如何解析动态域名
- 如何让 Raspberry 给自己发短信
- 如何监控摄像头

5.1 挂载磁盘

装完 RaspBian 系统后，TF 卡的容量也剩下不了多少了。巧妇难为无米之炊，想干点别的都会捉襟见肘，还是先给 Raspberry 挂载一个大容量的移动硬盘，或者是硬盘盒吧。

5.1.1 硬件准备

如果可以，尽可能选择带电源的移动硬盘或者是硬盘盒。Raspberry 的电源一般选择 5V+2A 足矣。但是想它带动大容量的移动硬盘就有点勉强了。所以尽量选择自带电源的大容量移动硬盘吧。将移动硬盘连好电源后，插入 Raspberry 的 USB 接口（Raspberry 的 USB 接口也可以作为电源输入口，但不建议这样做，还是让它们各司其职比较好）。

5.1.2 软件设置

启动 Raspberry，使用 Putty 远程登录 Raspberry，如图 5-1 所示。

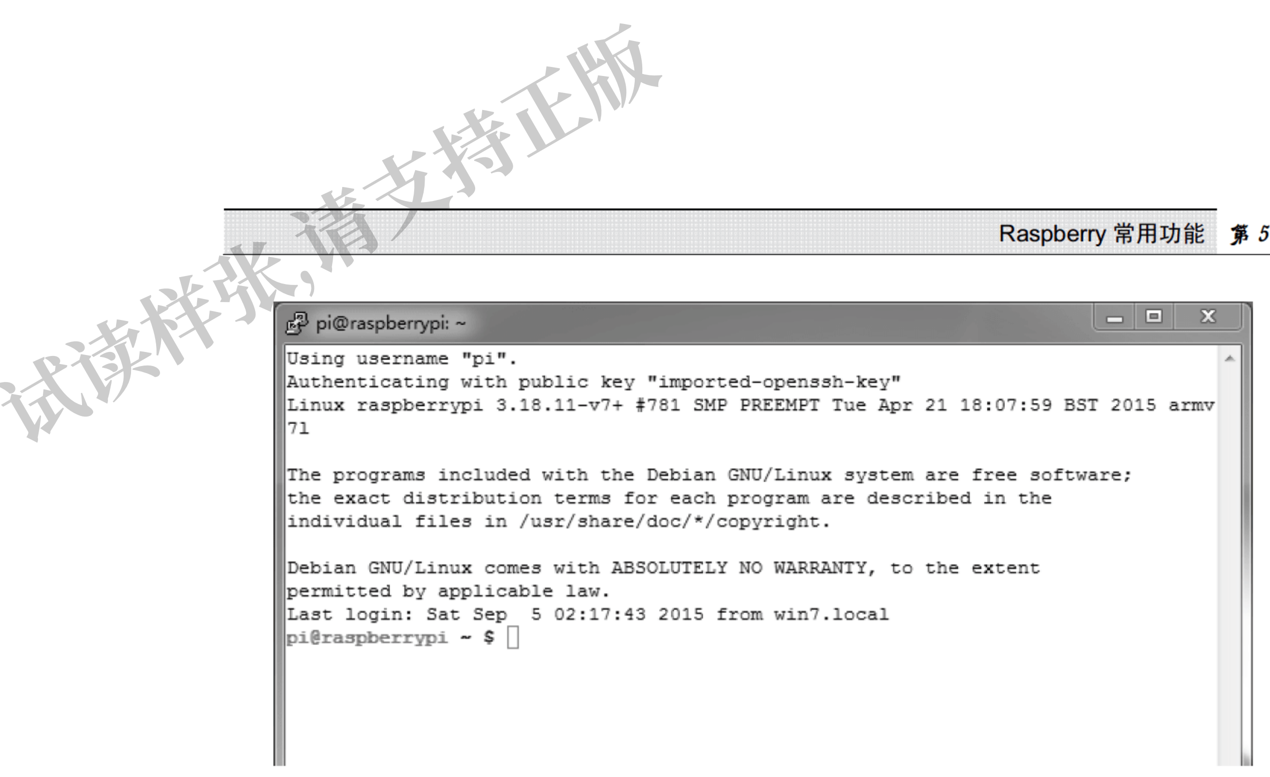


图 5-1　Putty 登录 Raspberry

1. 命令简介

先来学习 fdisk 命令。用 man fdisk 看看，如图 5-2 所示。

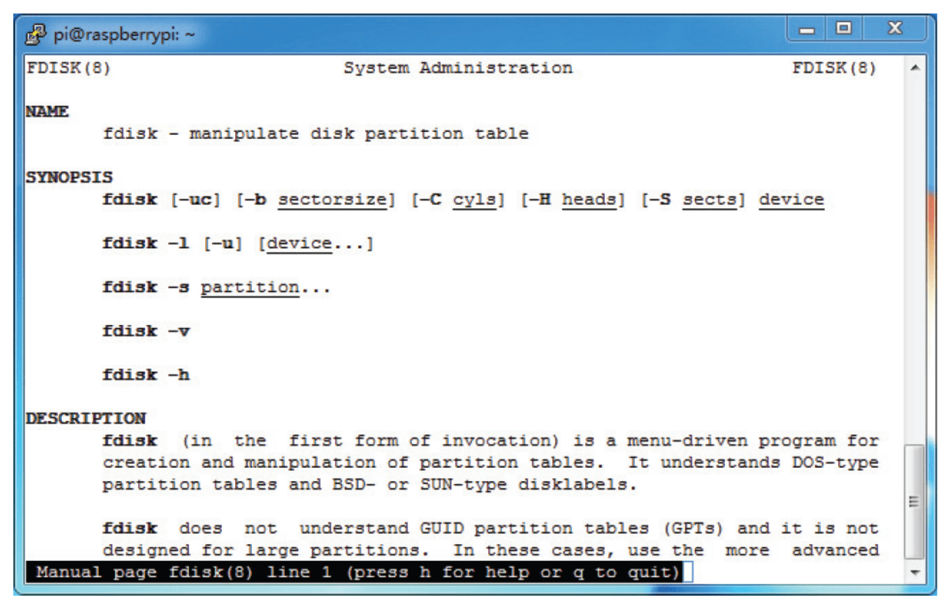


图 5-2　man fdisk

2. 检测硬盘，确定目标

fdisk 是 Linux 下管理磁盘的工具。它的功能非常强大，完全不逊于大名鼎鼎的 PQmagic。fdisk 可以对磁盘进行添加、删除、转换等等。几乎每个 Linux 发行版本都会默认安装 fdisk。缺点就是没有 GUI 界面，只能在命令行下操作，不太友好，如图 5-3 所示。执行命令：

```
sudo fdisk -l
```

```
pi@raspberrypi ~ $ sudo fdisk -l

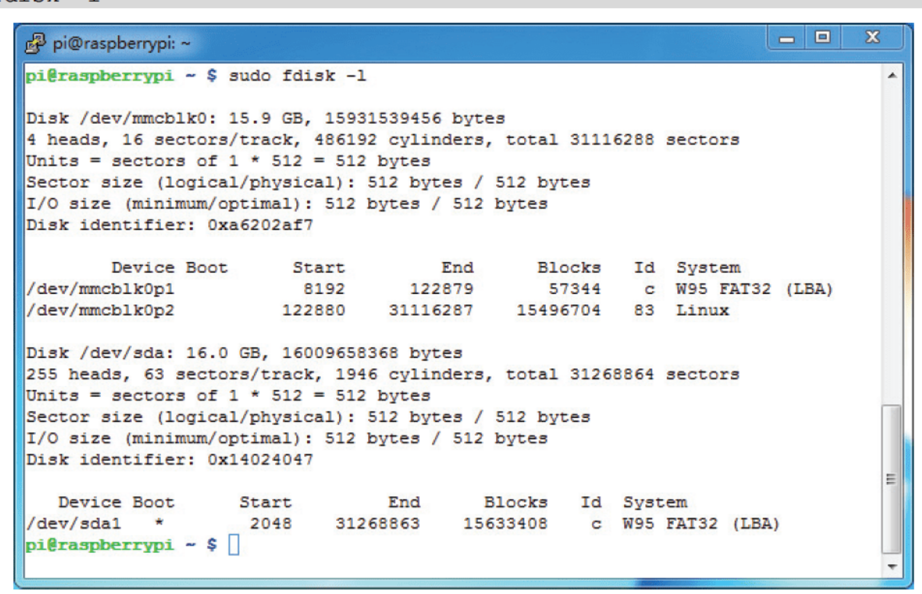
Disk /dev/mmcblk0: 15.9 GB, 15931539456 bytes
4 heads, 16 sectors/track, 486192 cylinders, total 31116288 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xa6202af7

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880    31116287    15496704   83  Linux
pi@raspberrypi ~ $
```

图 5-3　fdisk -l

将大容量移动硬盘插入 Raspberry 的 USB 接口。再次执行命令，如图 5-4 所示。

```
sudo fdisk -l
```



```
pi@raspberrypi ~ $ sudo fdisk -l

Disk /dev/mmcblk0: 15.9 GB, 15931539456 bytes
4 heads, 16 sectors/track, 486192 cylinders, total 31116288 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0xa6202af7

        Device Boot      Start         End      Blocks   Id  System
/dev/mmcblk0p1            8192      122879       57344    c  W95 FAT32 (LBA)
/dev/mmcblk0p2          122880    31116287    15496704   83  Linux

Disk /dev/sda: 16.0 GB, 16009658368 bytes
255 heads, 63 sectors/track, 1946 cylinders, total 31268864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x14024047

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    31268863    15633408    c  W95 FAT32 (LBA)
pi@raspberrypi ~ $
```

图 5-4　fdisk 显示磁盘

从上图中可以看出多出了一个/dev/sda 的设备，容量是 16GB（这里只是用 U 盘做测试，实际应用时可以使用 500GB，甚至更大的移动硬盘）。

3. 使用 fdisk 磁盘分区

下面开始对这个添加的移动硬盘进行分区、转换、格式化。执行命令：

```
sudo fdisk /dev/sda
```

使用 fdisk 命令，对设备/dev/sda（刚插入 USB 的设备）进行操作。是/dev/sda 不是/dev/sda1。要说明的是在 Linux 下插入的磁盘是 IDE 接口的，系统默认它的名字是/dev/hda、/dev/hdb……如果磁盘是 STAT 接口的，将被认为是/dev/sda、/dev/sdb……USB 磁盘也被认为是/dev/sd*。/dev/sda 是

插入的磁盘设备，而/dev/sda1 则是磁盘设备的第一个分区，那么第二个分区毫无疑问就是/dev/sda2 了，依次类推。所以，要操作整块磁盘，应该执行的命令是 sudo fdisk /dev/sda，而不是/dev/sda1。如图 5-5 所示。

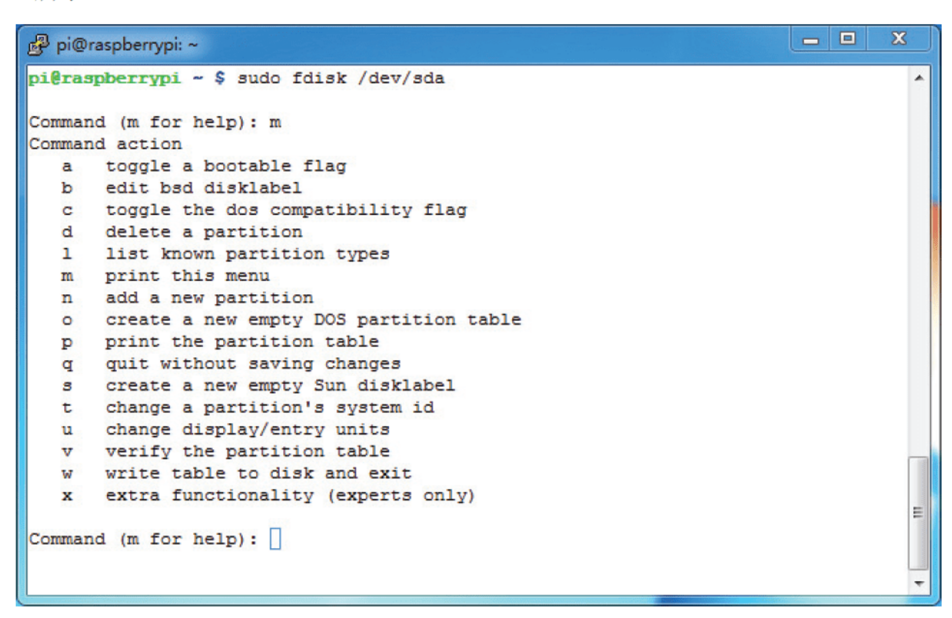


图 5-5　sudo fdisk /dev/sda

来看一下这些命令的功能（常用的注有中文，其他的功能不常用）。

- a：toggle a bootable flag
- b：edit bsd disklabel，编辑 bsd 磁盘列表
- c：toggle the dos compatibility flag
- d：delete a partition，删除分区
- l：list known partition types，列出分区的类型
- m：print this menu，显示帮助菜单
- n：add a new partition，添加一个新分区
- o：create a new empty DOS partition table，创建一个新的 DOS 磁盘列表
- p：print the partition table，列出现有的分区列表
- q：quit without saving changes，不保存修改，退出。
- s：create a new empty Sun disklabel，创建一个新的 Sun 磁盘列表
- t：change a partition's system id，修改磁盘分区类型（fat32，ext3……）
- u：change display/entry units
- v：verify the partition table
- w：write table to disk and exit，写入磁盘列表，保存退出。
- x：extra functionality （experts only），扩展应用，专家模式。

选择 p 查看现有的分区列表，如图 5-6 所示。

```
Command (m for help): p

Disk /dev/sda: 16.0 GB, 16009658368 bytes
255 heads, 63 sectors/track, 1946 cylinders, total 31268864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x14024047

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    31268863    15633408    c  W95 FAT32 (LBA)

Command (m for help):
```

图 5-6　fdisk 显示分区

可以看到已经有一个分区/dev/sda1 了。下面使用 fdisk 完整地将磁盘操作一次。首先是删除现有的磁盘，选择 d，如图 5-7 所示。

```
Command (m for help): p

Disk /dev/sda: 16.0 GB, 16009658368 bytes
255 heads, 63 sectors/track, 1946 cylinders, total 31268864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x14024047

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *        2048    31268863    15633408    c  W95 FAT32 (LBA)

Command (m for help): d
Selected partition 1
```

图 5-7　fdisk 删除分区

因为只有一个磁盘分区，所以就无须选择删除哪个分区了。如果有多个分区，则先选择 d 进行删除，然后再来选择删除哪个分区。一步步将所有分区都删除掉。再输入 p 显示现有的分区。如图 5-8 所示。

```
Command (m for help): p

Disk /dev/sda: 16.0 GB, 16009658368 bytes
255 heads, 63 sectors/track, 1946 cylinders, total 31268864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x14024047

   Device Boot      Start         End      Blocks   Id  System

Command (m for help):
```

图 5-8　fdisk 显示分区

现在显示磁盘没有分区了。下一步给磁盘划分新的分区，选择 n，如图 5-9 所示。



```
Command (m for help): p

Disk /dev/sda: 16.0 GB, 16009658368 bytes
255 heads, 63 sectors/track, 1946 cylinders, total 31268864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x14024047

   Device Boot      Start         End      Blocks   Id  System

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1):
Using default value 1
First sector (2048-31268863, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-31268863, default 31268863):
Using default value 31268863

Command (m for help):
```

图 5-9　fdisk 新建分区

fdisk 提示选择新分区是主分区还是扩展分区，主分区选择 p，扩展分区选择 e，默认选择是主分区。选择完毕后，再选择分区的起始位置，默认是 2048MB 的位置开始，按 Enter 键确认。再选择分区的结束位置。默认是磁盘末尾，也就是整块磁盘，按 Enter 键确认。如果只需要一个分区，到这里就可以了。

如果想分多个分区，则输入磁盘的大小。例如需要的新分区是 10GB，则输入+10GB；如果需要新分区是 100MB，则输入+100MB；然后再次输入 n，为第二个分区选择主分区，扩展分区，分区大小等等。

在这里，这块磁盘只是用来存储文件，弥补 Raspberry 磁盘容量的不足，所以只需要一个分区就可以了。输入 p 显示现在的磁盘列表，如图 5-10 所示。

```
Command (m for help): p

Disk /dev/sda: 16.0 GB, 16009658368 bytes
255 heads, 63 sectors/track, 1946 cylinders, total 31268864 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x14024047

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1            2048    31268863    15633408   83  Linux

Command (m for help):
```

图 5-10　fdisk 显示分区

看新分区的 Id 项，显示的是 83。现在输入 l，列出分区的类型，来看一下 83 代表什么，如图 5-11 所示。

```
Command (m for help): l

 0  Empty           24  NEC DOS         81  Minix / old Lin bf  Solaris
 1  FAT12           27  Hidden NTFS Win 82  Linux swap / So c1  DRDOS/sec (FAT-
 2  XENIX root      39  Plan 9          83  Linux           c4  DRDOS/sec (FAT-
 3  XENIX usr       3c  PartitionMagic  84  OS/2 hidden C:  c6  DRDOS/sec (FAT-
 4  FAT16 <32M      40  Venix 80286     85  Linux extended  c7  Syrinx
 5  Extended        41  PPC PReP Boot   86  NTFS volume set da  Non-FS data
 6  FAT16           42  SFS             87  NTFS volume set db  CP/M / CTOS / .
 7  HPFS/NTFS/exFAT 4d  QNX4.x          88  Linux plaintext de  Dell Utility
 8  AIX             4e  QNX4.x 2nd part 8e  Linux LVM       df  BootIt
 9  AIX bootable    4f  QNX4.x 3rd part 93  Amoeba          e1  DOS access
 a  OS/2 Boot Manag 50  OnTrack DM      94  Amoeba BBT      e3  DOS R/O
 b  W95 FAT32       51  OnTrack DM6 Aux 9f  BSD/OS          e4  SpeedStor
 c  W95 FAT32 (LBA) 52  CP/M            a0  IBM Thinkpad hi eb  BeOS fs
 e  W95 FAT16 (LBA) 53  OnTrack DM6 Aux a5  FreeBSD         ee  GPT
 f  W95 Ext'd (LBA) 54  OnTrackDM6      a6  OpenBSD         ef  EFI (FAT-12/16/
10  OPUS            55  EZ-Drive        a7  NeXTSTEP        f0  Linux/PA-RISC b
11  Hidden FAT12    56  Golden Bow      a8  Darwin UFS      f1  SpeedStor
12  Compaq diagnost 5c  Priam Edisk     a9  NetBSD          f4  SpeedStor
14  Hidden FAT16 <3 61  SpeedStor       ab  Darwin boot     f2  DOS secondary
16  Hidden FAT16    63  GNU HURD or Sys af  HFS / HFS+      fb  VMware VMFS
17  Hidden HPFS/NTF 64  Novell Netware  b7  BSDI fs         fc  VMware VMKCORE
18  AST SmartSleep  65  Novell Netware  b8  BSDI swap       fd  Linux raid auto
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 75  PC/IX           be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1 80  Old Minix

Command (m for help):
```

图 5-11 fdisk 分区格式列表

从图中可以看出 83 代表的是 Linux 分区。如果该移动硬盘只在 Raspberry 上使用，无须更改，这样就可以了。再选择 w 保存退出就可以了。

如果该移动硬盘还需要放到 Windows 上使用，Linux 分区在 Windows 中是无法直接识别的，那就需要选择 t，改变分区类型，如图 5-12 所示。

```
1b  Hidden W95 FAT3 70  DiskSecure Mult bb  Boot Wizard hid fe  LANstep
1c  Hidden W95 FAT3 75  PC/IX           be  Solaris boot    ff  BBT
1e  Hidden W95 FAT1 80  Old Minix

Command (m for help): t
Selected partition 1
Hex code (type L to list codes):
```

图 5-12 fdisk 修改分区格式

再输入 b，选择 win95 fat32 类型。再选择 w 保存退出。这样 fat32 格式可以被 Windows 和 Linux 识别。我在这里选择是的 83，仅在 Linux 下使用。



注 意

fdisk 是非常优秀的分区工具。除了没有图形界面这个不是缺点的缺点外，它远比明星软件 PQ 强大。如果在使用 PQ 无法解决问题，那就使用 Linux 启动，挂载硬盘后使用 fdisk 吧。相信它是不会让人失望的。

4. 格式化磁盘

分区完毕后，使用 mkfs 命令对新分区格式化。如图 5-13 所示，执行命令：

```
sudo mkfs.ext4 /dev/sda1
```

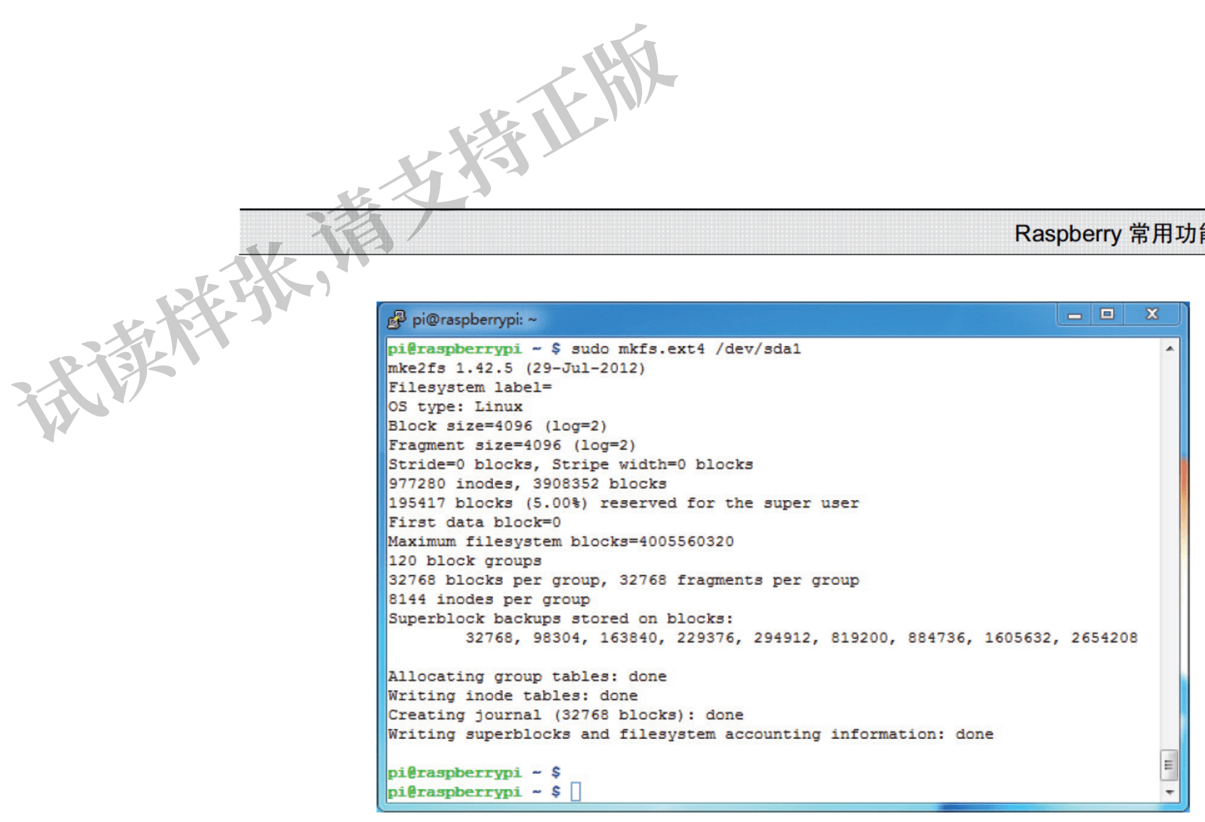


图 5-13　mkfs 格式化磁盘

ext4 是一种 Linux 的分区格式。当然 Linux 还有其他的几种分区格式，但是最流行的还是 ext。如果没有特殊要求，选择 ext4 是最安全的做法。

5. 挂载磁盘到系统

磁盘已经分区格式化完毕，但还不能使用。系统虽然已经识别了磁盘分区，但还没有把磁盘分区加入到系统中去。最后要把磁盘分区挂载到系统中去。

Linux 系统中，挂载磁盘一般是在/mnt 或/media 目录中。执行命令：

```
sudo mkdir /mnt/disk
sudo mount -t ext4 -o user,defaults /dev/sda1 /mnt/disk
```

好了，现在已经把磁盘分区挂载到/mnt/disk 目录中了。但这种挂载是临时的挂载，Raspberry 重启后，又得重新执行挂载命令。修改/etc/fstab 文件，一劳永逸解决这个问题。使用 vi /etc/fstab，在/etc/fstab 的最后加上一行：

```
/dev/sda1         /mnt/disk       ext4         defaults,user       0 0
```

6. 测试挂载

现在已经将磁盘完全挂载到 Raspberry 上了。此时/mnt/disk 的属主是 root，将它修改成 pi，以便与 pi 用户也可以读写。如图 5-14 所示，执行命令：

```
sudo chown pi:pi /mnt/disk
ls -l /mnt/disk
touch /mnt/disk/abc.txt
ls -l /mnt/disk/abc.txt
rm /mnt/disk/abc.txt
```

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ sudo chown pi:pi /mnt/disk
pi@raspberrypi ~ $ ls -l /mnt/disk/
total 16
drwx------ 2 root root 16384 Sep  7 14:58 lost+found
pi@raspberrypi ~ $ touch /mnt/disk/abc.txt
pi@raspberrypi ~ $ ls -l /mnt/disk/abc.txt
-rw-r--r-- 1 pi pi 0 Sep  8 10:15 /mnt/disk/abc.txt
pi@raspberrypi ~ $ rm /mnt/disk/abc.txt
pi@raspberrypi ~ $
```

图 5-14　添加 pi 用户权限

现在 pi 用户也可以自由地读写新添加的硬盘了。Raspberry 的空间已经扩容完毕。想扩展 Raspberry 的功能，干点别的吗？Do it。

5.2　Aria2 下载机

Aria2 是一个命令行下运行、多协议、多来源、多平台下载工具（HTTP/HTTPS、FTP、BitTorrent、Metalink），比 Linux 下默认的下载工具 Wget 更加强大，如果只是将它单纯地作为一个命令行下载工具，那就有点浪费了。它的缺点是 Linux 下软件的通病，没有漂亮的 GUI。不过没关系，配合 yaaw 作为 Web 前台，可以将 Raspberry 改造成一台下载机。

5.2.1　安装下载组件

Aria2 下载机功能强大，但需要的组件不多，只要安装 Aria2 和 yaaw 即可，再配合 Web 服务器（这里我用的是 Apache2）就组合成了功能强大的下载机。先来下载这两个软件。

Aria2 已经在 Debian 的官方源里了。所以只需要执行命令：

```
sudo apt-get instrall aria2
```

安装就是这么简单，来看看 man aria2c，如图 5-15 所示。

```
pi@raspberrypi: ~
ARIA2C(1)                          aria2                          ARIA2C(1)

NAME
       aria2c - The ultra fast download utility

SYNOPSIS
       aria2c [<OPTIONS>] [<URI>|<MAGNET>|<TORRENT_FILE>|<METALINK_FILE>] ...

DESCRIPTION
       aria2  is  a utility for downloading files. The supported protocols are
       HTTP(S), FTP, BitTorrent, and Metalink. aria2 can download a file  from
       multiple  sources/protocols  and tries to utilize your maximum download
       bandwidth. It supports downloading a file from HTTP(S)/FTP and  BitTor‐
       rent  at  the  same time, while the data downloaded from HTTP(S)/FTP is
       uploaded to the BitTorrent swarm.  Using  Metalink's  chunk  checksums,
       aria2  automatically  validates chunks of data while downloading a file
       like BitTorrent.

OPTIONS
    Basic Options
       -d, --dir=<DIR>
              The directory to store the downloaded file.
 Manual page aria2c(1) line 1 (press h for help or q to quit)
```

图 5-15　man aria2

熟悉下 Aria2 常用的命令行参数：

- -v：--version，显示版本号
- -h：--help，显示帮助信息
- -l：--log=LOG，设置日志
- -d：--dir=DIR，设置存储下载文件的目录
- -o：--out=FILE，设置存储下载文件的文件名
- -s：--split=N，分段下载

基本上和 Wget 相差不大，比 Wget 强的方面是支持 JSON-RPC，所以可以使用 YAAW 作为它的 Web 前台。

YAAW 是 Yet Another Aria2 Web Frontend 的缩写。顾名思义，YAAW 完全是为了 Aria2 而开发的。YAAW 的首页/demo: http://github.com/binux/yaaw/、GitHub: https://github.com/binux/yaaw。

虽然 YAAW 非常不错，但都是 E 文，看起来还是很碍眼的。幸好有好心人汉化了 YAAW 放到了 Github 上了，所以想下载中文版本的 YAAW，请自行搜索一下。这里还是使用原版的 YAAW 输入命令：

```
cd /home/pi/www  #这个是前面 web 服务的主目录。
git clone https://github.com/binux/yaaw
```

YAAW 下载好了。现在使用浏览器打开 http://192.168.2.91:8080/yaaw （前面配置 Apache2 时设置的端口是 8080），如图 5-16 所示。

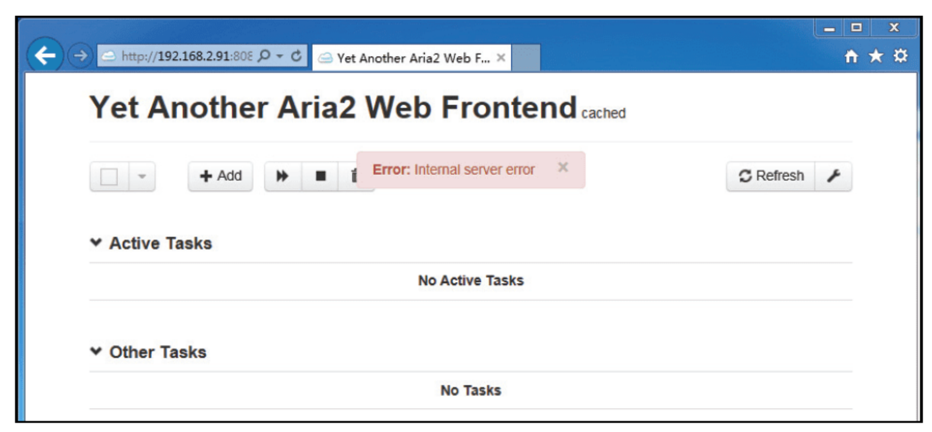


图 5-16　浏览 YAAW

提示有错误。没关系，那是没有配置好。配置一下就没问题了。

5.2.2　Aria2 配置

先编写 Aria2 的配置文件/etc/aria2.conf，系统本身没有这个文件，需要自行创建。执行命令：

```
sudo touch /etc/aria2.conf
sudo mkdir -pv /mnt/disk/download
```

```
touch /mnt/disk/download/.aria2.session
```

Aria2 的配置项很多，这里只需要最简单的方案。/etc/aria2.conf 的代码如下：

```
 1 dir=/mnt/disk/download
 2 disable-ipv6=true
 3 enable-rpc=true
 4 rpc-allow-origin-all=true
 5 rpc-listen-all=true
 6 rpc-listen-port=6800
 7 continue=true
 8 input-file=/mnt/disk/download/.aria2.session
 9 save-session=/mnt/disk/download/.aria2.session
10 max-concurrent-downloads=3
```

完了后运行命令：

```
aria2c --conf-path=/etc/aria2.conf
```

测试看看有没有错误，如果没有错误的话，按 Ctrl + C 组合键终止程序，继续下一步；如果有错误的话，会提示你 conf 文件哪里错误。

创建 aria2c，将 Aria2 做成系统的服务，让它随系统一起启动。

```
sudo vi /etc/init.d/aria2c
```

/etc/init.d/aria2c 的代码如下：

```
 1 #!/bin/sh
 2 ### BEGIN INIT INFO
 3 # Provides:          aria2
 4 # Required-Start:    $remote_fs $network
 5 # Required-Stop:     $remote_fs $network
 6 # Default-Start:     2 3 4 5
 7 # Default-Stop:      0 1 6
 8 # Short-Description: Aria2 Downloader
 9 ### END INIT INFO
10
11 case "$1" in
12 start)
13
14 echo  -n "Starting aria2c "
15 echo
16 aria2c --conf-path=/etc/aria2.conf -D
17 ;;
18 stop)
19
20 echo  -n "Shutting down aria2c "
21 echo
22 killall aria2c
23 ;;
```

```
24 restart)
25
26 echo  -n "restart aria2c "
27 echo
28 killall aria2c
29 aria2c --conf-path=/etc/aria2.conf -D
30 ;;
31 esac
32 exit
```

测试服务是否可以启动，然后将 Aria2 添加到系统启动服务中去，如图 5-17 所示。

```
pi@raspberrypi ~/www $ sudo /etc/init.d/aria2c start
Starting aria2c
pi@raspberrypi ~/www $ sudo update-rc.d aria2c defaults
update-rc.d: using dependency based boot sequencing
pi@raspberrypi ~/www $ sudo insserv aria2c
pi@raspberrypi ~/www $
```

图 5-17　添加启动服务

现在用浏览器打开 http://192.168.2.91:8080/yaaw，如图 5-18 所示。

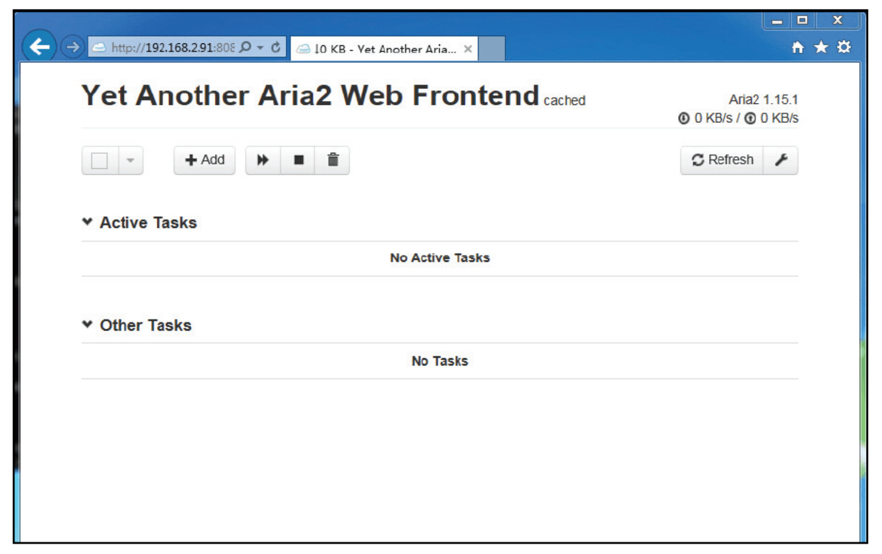


图 5-18　浏览 yaaw

Aria2 下载机安装配置完毕。

5.2.3　测试 Aria2 下载机

测试一下这个下载机，以下载 Debian 的 rom 为例。单击 Add 按钮，将 debian-live-8.2.0-amd64-gnome-desktop.iso 的下载地址输入 Upload Torrent 按钮前面的文本框内，在 File Name 后面的文本框内输入保存的文件名，如图 5-19 所示。

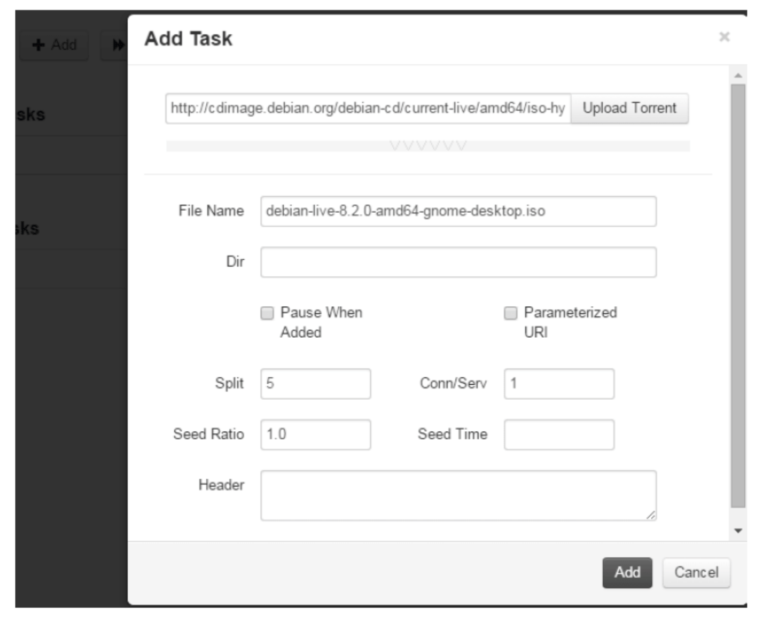


图 5-19　添加下载任务

单击 Add 按钮，开始任务下载。如图 5-20 所示。

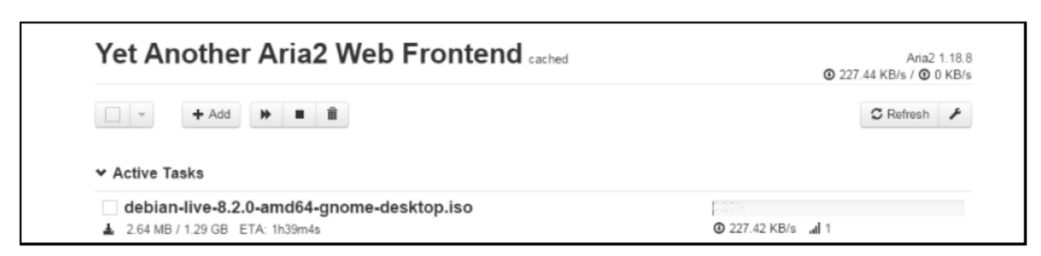


图 5-20　任务下载中

这个下载速度取决于宽带的速度和提供下载的服务器的速度。一般来说，国内的服务器是可以达到满带宽下载的。就算是国外的服务器不能满带宽下载也没关系。Raspberry 极其省电，完全可以 24×7 开机，输入下载的地址后就可以任其自行下载，绿色环保，安全方便。

好了，不要管它了，任其发挥，自由下载吧。实际上 yaaw 还可以配合迅雷离线下载，使用迅雷离线下载助手插件，这是个 Chrome 上的插件。它可以将迅雷离线下载好的文件通过 yaaw 下载回来。如果有国外站点文件下载速度比较慢，可以通过这种方法快速地将其下载回来。


注　意

Chrome 和 Firefox 都有相应的插件可配合迅雷离线使用 yaaw。这种方法唯一的缺陷就是必须是迅雷会员才能使用。

5.3　迅雷远程下载

既然说到了下载工具，就不得不提起迅雷。在国内迅雷可谓是当之无愧的行业大佬，可惜的

是迅雷没有合适的 Linux 版本。好在迅雷给 Linux 留下了一个小口子，迅雷远程下载。

5.3.1 下载迅雷远程下载固件

迅雷远程下载固件主要是为嵌入式设计的，既然可以用于嵌入式，比嵌入式更强大的 Raspberry 当然也可以用了。迅雷的固件的下载地址是 http://luyou.xunlei.com/thread-15167-1-1.html?_t=1443319618。Xware 为了适应多种设备，开发的版本很多，如图 5-21 所示。

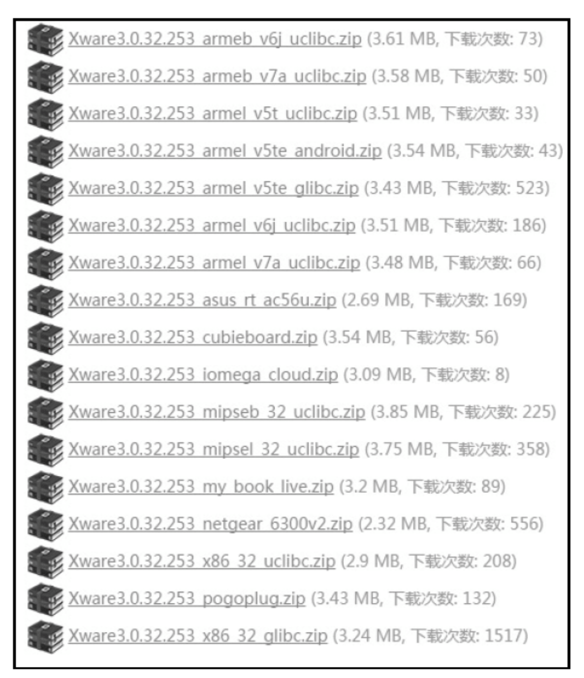


图 5-21 迅雷固件列表

应该挑选哪个版本的 Xware 呢？已有热心的网友做好了测试，只需要按图索骥就可以了。打开 http://g.xunlei.com/thread-208-1-1.html，然后在网页搜索 raspberry，如图 5-22 所示。

树莓派CPUINFO	Linux raspberrypi 3.10.24+ #614 PREEMPT Thu Dec 19 20:38:42 GMT 2013 armv6l GNU/Linux	ARMv6-compatible processor rev 7 (v6l)	xware_armel_v5te_glibc	√

图 5-22 迅雷固件选择

选择 Xware_armel_v5te_glibc 版本就可以了。也就是图 5-21 中的 Xware3.0.32.253_armel_v5te_glibc.zip。

5.3.2 设置迅雷远程下载

已经取得了相应的固件，下面开始安装。

1. 解压固件

将 Xware3.0.32.253_armel_v5te_glibc.zip 下载到 pi 用户的主目录下。执行命令得到的结果如图 5-23 所示。

```
mkdir xunlei
mv Xware3.0.32.253_armel_v5te_glibc.zip ./xunlei/
cd xunlei/
unzip Xware3.0.32.253_armel_v5te_glibc.zip
```

```
pi@raspberrypi: ~/xunlei
pi@raspberrypi ~ $ ls
total 3540
drwxr-xr-x 4 pi pi    4096 2015/08/07 20:48:56 code
drwxr-xr-x 2 pi pi    4096 2015/07/10 01:19:13 Desktop
drwxr-xr-x 2 pi pi    4096 2015/09/01 20:12:57 ftp
drwxr-xr-x 2 pi pi    4096 2015/09/19 00:25:28 log
drwxr-xr-x 4 pi pi    4096 2015/09/21 02:49:28 www
-rwxr--r-- 1 pi pi 3600405 2015/09/27 10:42:28 Xware3.0.32.253_armel_v5te_glibc.
zip
pi@raspberrypi ~ $ mkdir xunlei
pi@raspberrypi ~ $ mv Xware3.0.32.253_armel_v5te_glibc.zip ./xunlei/
pi@raspberrypi ~ $ cd xunlei/
pi@raspberrypi ~/xunlei $ unzip Xware3.0.32.253_armel_v5te_glibc.zip
Archive:  Xware3.0.32.253_armel_v5te_glibc.zip
  inflating: vod_httpserver
  inflating: log.ini
  inflating: thunder_mounts.cfg
  inflating: etm_monitor
  inflating: etm_xware
  inflating: xware_csh.sh
  inflating: xware_bash.sh
pi@raspberrypi ~/xunlei $
```

图 5-23　安装前准备

2. 修改配置文件

查看迅雷固件的配置文件，如图 5-24 所示。

```
pi@raspberrypi: ~/xunlei
pi@raspberrypi ~/xunlei $ ls *.cfg
-rwxr-xr-x 1 pi pi 279 2015/08/06 17:36:34 thunder_mounts.cfg
pi@raspberrypi ~/xunlei $ cat thunder_mounts.cfg
#过滤与第二列相同的挂载路径
invalid_mounts
{
        rootfs /
}

#仅接受以下列路径开头的挂载路径
available_mounts
{
        /tmp/usbmounts/
        /tmp/HDD/
        /mnt/USB/
        /mnt/HardDisk/
}

#下列目录被认为是分区，并在程序运行期间不变
virtual_mounts
{
}
pi@raspberrypi ~/xunlei $
```

图 5-24　配置文件

修改这个配置文件，将 /mnt/disk/download/ 加入配置文件，修改后的配置文件

thunder_mounts.cfg 代码如下：

```
 1 #过滤与第二列相同的挂载路径
 2 invalid_mounts
 3 {
 4     rootfs /
 5 }
 6
 7 #仅接受以下列路径开头的挂载路径
 8 available mounts
 9 {
10     /tmp/usbmounts/
11     /tmp/HDD/
12     /mnt/USB/
13     /mnt/HardDisk/
14 ###这里是自行添加的下载目录
15     /mnt/disk/
16 }
17
18 #下列目录被认为是分区，并在程序运行期间不变
19 virtual_mounts
20 {
21 }
```

3. 获取激活码

配置文件修改完毕后，开始安装迅雷远程下载，执行命令得到的结果如图 5-25 所示。

```
sh xware_bash.sh
```

```
pi@raspberrypi: ~/xunlei
executing ./etm_xware  --system_path=. --disk_cfg=./thunder_mounts.cfg --etm_cfg
=./etm.ini --log_cfg=./log.ini --pid_file=./xunlei.pid --license=150806200100000
4u0000007vub32aatrnhgtpuwdz --import_v1v2_mode=2 --listen_addr=0.0.0.0:19000 --h
ubble_report_pipe_path=./etm_hubble_report.pipe
get net address result:0, adapter:eth0, mac:B827EB650DA2, ipv4:5b02a8c0, brdaddr
:ff02a8c0, netmask:ffffff
preferred_network_adapter:eth0, mac:B827EB650DA2, ipv4:5b02a8c0, brdaddr:ff02a8c
0, netmask:ffffff
get net address result:0, adapter:wlan0, mac:E84E0613192B, ipv4:7d02a8c0, brdadd
r:ff02a8c0, netmask:ffffff
fopen [./etm.ini] fail:No such file or directory.
setting_load_cfg ./etm.ini fail:2
Bind error: 98

et version:et_3.0.0_25
settings_config_load, cfg_file_name = /home/pi/xunlei/download.cfg
get net address result:0, adapter:eth0, mac:B827EB650DA2, ipv4:5b02a8c0, brdaddr
:ff02a8c0, netmask:ffffff
preferred_network_adapter:eth0, mac:B827EB650DA2, ipv4:5b02a8c0, brdaddr:ff02a8c
0, netmask:ffffff
get net address result:0, adapter:wlan0, mac:E84E0613192B, ipv4:7d02a8c0, brdadd
r:ff02a8c0, netmask:ffffff
active_key:
```

图 5-25　取得激活码

4. 绑定激活码

这里要留意的是下载目录的配置文件和激活码。浏览器打开 http://yuancheng.xunlei.com，输入用户名，密码登录，如图 5-26 所示。



图 5-26　登录迅雷远程

在“绑定”前面的文本框中输入刚才得到的 6 位数激活码，单击“绑定”，如图 5-27 所示。

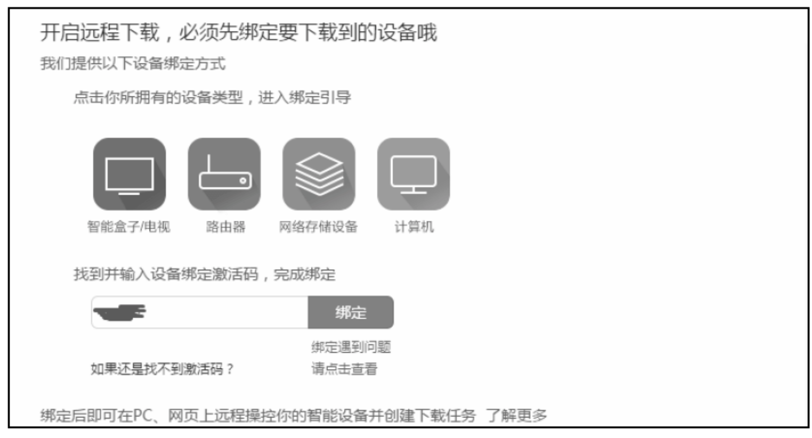


图 5-27　绑定激活码

完成后，单击铅笔图形的编辑按钮，修改设备名，如图 5-28 所示。



图 5-28　修改设备名

修改完毕后单击“确定”按钮，如图 5-29 所示。



图 5-29　保存修改设备名

好了，现在可以使用迅雷远程下载了。打开迅雷登录，单击远程设备，得到刚才添加的远程设备，如图 5-30 所示。



图 5-30　显示远程设备

图片中显示的下载目录 C:/TDDOWNLOAD 文件夹其实就是 Raspberry 的 /mnt/disk/TDDOWNLOAD 文件夹，需要自行创建，执行命令：

```
sudo mkdir -pv /mnt/disk/TDDOWNLOAD
```

最后，将远程设备的启动命令添加到/etc/rc.local 中去，执行命令：

```
sudo vi /etc/rc.local
```

修改完毕后，结果如图 5-31 所示。

```
pi@raspberrypi: ~/xunlei
pi@raspberrypi ~/xunlei $ tail /etc/rc.local
# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

exit 0

#############
sh /home/pi/xunlei/xware_bash.sh
pi@raspberrypi ~/xunlei $
```

图 5-31　添加到开机启动

大功告成，以后启动 Raspberry 后，就可以直接用迅雷远程下载下载文件了。这里要注意的是，外挂的磁盘必须挂载到刚才修改的配置文件 thunder_mounts.cfg 中设定的几个目录。可以是系统本身默认的/mnt/USB/，也可以是/mnt/HardDisk/，当然自行添加的/mnt/disk 更没问题（在 6.1 节中已经将磁盘挂载到/mnt/disk 了，就无须在此再挂载一次了）



注 意

迅雷远程固件可以在 Raspberry 上使用，也支持 x86_32 的 PC，就是不支持 x86_64。

5.4　动态域名解析

做好了服务器，如果只能在内网使用，那未免也太无趣了。独乐乐，与人乐乐，孰乐？好东西要分享。本章将把建立好的 Web 服务器放到公网上去，让大家可以通过公网域名访问放在 Raspberry 上的个人服务器。

5.4.1　神器花生壳

一般网络服务商分配给人的都是动态的 IP，也就是过一段时间就变换一次 IP 地址。这样一来，如果想将域名与 IP 对应起来，就不得不使用动态域名解析。而国内最好用的动态域名解析软件就是花生壳。

花生壳是一个动态域名解析软件。利用花生壳无论在任何地点、任何时间、使用任何线路，均可利用花生壳建立拥有固定域名和最大自主权的互联网主机。“花生壳动态域名解析软件”支持的线路包括普通电话线、ISDN、ADSL、有线电视网络、双绞线到户的宽带网和其他任何能够提供互联网真实 IP 的接入服务线路，而无论连接获得的 IP 属于动态还是静态，甚至服务器藏于内网，都可以使用花生壳穿透内网。

5.4.2　下载安装花生壳

花生壳在 Windows 下的客户端就不说了，那必定是丰富多彩的。在 Linux 下也有客户端，可是最新的客户端不支持 Raspberry，所以只能下载老版本的花生壳。执行命令如图 5-32 所示。

```
wget http://download.oray.com/peanuthull/phddns-2.0.2.16556.tar.gz
```

```
tar zxvf phddns-2.0.2.16556.tar.gz
cd phddns-2.0.2.16556
```

```
pi@raspberrypi: ~/phddns-2.0.2.16556
pi@raspberrypi ~/phddns-2.0.2.16556 $ ls
aclocal.m4  configure     depcomp     Makefile.am  NEWS
AUTHORS     configure.in  INSTALL     Makefile.in  README
ChangeLog   COPYING       install-sh  missing      src
pi@raspberrypi ~/phddns-2.0.2.16556 $
```

图 5-32　下载花生壳

编译安装花生壳，执行命令结果如图 5-33 所示。

```
aclocal\
autoconf
automake
./configure
make -j2
```

```
pi@raspberrypi: ~/phddns-2.0.2.16556
gcc -DPACKAGE_NAME=\"FULL-PACKAGE-NAME\" -DPACKAGE_TARNAME=\"full-package-name\"
 -DPACKAGE_VERSION=\"VERSION\" -DPACKAGE_STRING=\"FULL-PACKAGE-NAME\ VERSION\" -
DPACKAGE_BUGREPORT=\"BUG-REPORT-ADDRESS\" -DPACKAGE_URL=\"\" -DPACKAGE=\"phddns\
" -DVERSION=\"2.0.2.16556\" -DSTDC_HEADERS=1 -DHAVE_SYS_TYPES_H=1 -DHAVE_SYS_STA
T_H=1 -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -DHAVE_MEMORY_H=1 -DHAVE_STRINGS_H=1 -
DHAVE_INTTYPES_H=1 -DHAVE_STDINT_H=1 -DHAVE_UNISTD_H=1 -DHAVE_ARPA_INET_H=1 -DHA
VE_FCNTL_H=1 -DHAVE_MEMORY_H=1 -DHAVE_NETDB_H=1 -DHAVE_NETINET_IN_H=1 -DHAVE_STD
DEF_H=1 -DHAVE_STDLIB_H=1 -DHAVE_STRING_H=1 -DHAVE_SYS_IOCTL_H=1 -DHAVE_SYS_SOCK
ET_H=1 -DHAVE_SYS_TIME_H=1 -DHAVE_SYS_TIMEB_H=1 -DHAVE_TERMIOS_H=1 -DHAVE_UNISTD
_H=1 -DHAVE__BOOL=1 -DHAVE_STDBOOL_H=1 -DTIME_WITH_SYS_TIME=1 -DHAVE_SYS_SELECT_
H=1 -DHAVE_SYS_SOCKET_H=1 -DSELECT_TYPE_ARG1=int -DSELECT_TYPE_ARG234=\(fd_set\
\*\) -DSELECT_TYPE_ARG5=\(struct\ timeval\ \*\) -DRETSIGTYPE=void -DHAVE_STRFTIM
E=1 -DHAVE_VPRINTF=1 -DHAVE_GETHOSTBYNAME=1 -DHAVE_INET_NTOA=1 -DHAVE_ISASCII=1
-DHAVE_MEMSET=1 -DHAVE_SELECT=1 -DHAVE_SOCKET=1 -DHAVE_STRCHR=1 -DHAVE_STRSTR=1
-I. -I.    -g -O2 -MT phruncall.o -MD -MP -MF .deps/phruncall.Tpo -c -o phruncal
l.o phruncall.c
mv -f .deps/phruncall.Tpo .deps/phruncall.Po
gcc  -g -O2 -s  -o phddns base64.o blowfish.o generate.o log.o main.o md5.o phgl
obal.o phsocket.o phupdate.o phruncall.o
make[1]: Leaving directory '/home/pi/phddns-2.0.2.16556/src'
make[1]: Entering directory '/home/pi/phddns-2.0.2.16556'
make[1]: Nothing to be done for 'all-am'.
make[1]: Leaving directory '/home/pi/phddns-2.0.2.16556'
pi@raspberrypi ~/phddns-2.0.2.16556 $
```

图 5-33　安装花生壳

现在已成功地将花生壳安装到了/home/pi/phddns-2.0.2.16556/src 下了。

5.4.3　设置花生壳

使用花生壳前必须在花生壳的官网注册了账号，至少有一个域名，不管是免费域名还是购买的域名。

1. 配置花生壳

执行命令：

```
cd /home/pi/phddns-2.0.2.16556/src
mkdir /home/pi/etc
```

```
mkdir /home/pi/bin
./phddns
输入服务器地址，如无特殊情况可使用默认值
Enter server address（press ENTER use phLinux3.oray.net）:

输入您的 Oray 账号名称
Enter your Oray account:yourname

对应的 Oray 账号密码
Password:**********

选择绑定的网卡，如无特殊，默认即可
Network interface（s）:
eth0:192.168.2.91
lo:127.0.0.1
Choose one（default eth0）:eth0

选择日志保存到哪个文件
Log to use（default /var/log/phddns.log）:/home/pi/log/phddns.log

保存配置文件，选择 yes 则直接保存到/etc/phLinux.conf，输入 other 可以指定文件
Save to configuration file （/etc/phLinux.conf）?（yes/no/other）:other
Enter configuration filename（/etc/phLinux.conf）:/home/pi/etc/phLinux.conf

接下来程序将以交互模式开始运行
192.168.2.91
NIC bind success
OnStatusChanged okConnecting
OnStatusChanged okDomainListed
OnDomainRegistered skyvense22.gicp.net
OnStatusChanged okDomainsRegistered
UserType: 0
看到上面这些就表示登录成功，这个时候可以按 Ctrl+C 组合键先退出程序
```

执行命令：

```
cp /home/pi/phddns-2.0.2.16556/src/phddns /home/pi/bin/
/home/pi/bin/phddns -c /home/pi/etc/phLinux.conf -d
sudo echo "/home/pi/bin/phddns -c /home/pi/etc/phLinux.conf -d" >> /etc/rc.local
```

现在已经将本机绑定到了花生壳账户上了。然后登录花生壳官网，查看域名管理，如图 5-34 所示。



图 5-34　花生壳域名管理

2. 验证花生壳

绑定了花生壳后，验证是否绑定成功。在浏览器里输入注册的域名，如图 5-35 所示。

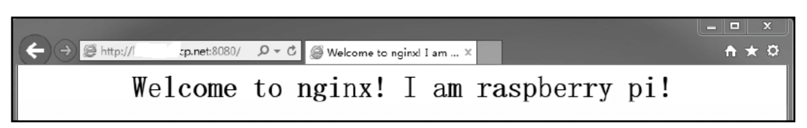


图 5-35　通过域名访问站点

花生壳的效果验证完毕。动态域名解析软件还有很多，花生壳是最常见的也是最方便的，如果有免费域名，用花生壳是最方便的。而且它可以直接穿透内网，无须再进行端口映射，省了很多的麻烦。


注　意

目前国内市面上流行的路由器基本上都原生态地支持花生壳。也就是说可以直接在路由器上登录花生壳账号。如果使用路由器登录花生壳，就必须使用端口映射将服务器的端口映射到路由器上。

5.5　无域名访问内网

花生壳是很好，但有人没有花生壳的免费域名怎么办？也很简单，只要找到本地的公网 IP，使用公网 IP 也行，只要做一个端口映射就可以了。

5.5.1　确定公网 IP

如何确定自己的公网 IP？通常我的做法就是打开 www.baidu.com，然后在搜索文本框中输入 IP，单击“百度一下”后，直接看第一个搜索结果就是了。仔细看一下，得到的这个 IP 是由 www.ip138.com 返回的。打开 www.ip138.com，查看它的源代码，发现这个 IP 地址是由 http://1111.ip138.com/ic.asp 返回的。

行了，思路出来了。直接用 python 脚本去访问 1111.ip138.com/ic.asp 这个网页，然后在返回的结果中查找需要的 IP 就可以了。使用 Putty 登录 Raspberry，执行命令：

```
mkdir -pv code/python/getNip
```

```
cd $_
touch2py getNip.py
vi getNip.py
```

最终的 getNip.py 的代码如下：

```
 1 #!/usr/bin/env python
 2 # -*- coding:utf-8 -*-
 3 #Author  :hstking
 4 #E-mail  :hstking@hotmail.com
 5 #Ctime   :2015/08/27
 6 #Mtime   :
 7 #Version :
 8
 9 import urllib2
10 import re
11 import os
12
13
14 ####  定义 GetNip 类
15 class GetNip():
16 ####  定义构造函数，可以用于定义类变量
17    def __init__(self):
18        self.logPath = os.path.expanduser('~') + os.sep + 'log'
19        self.nipFile = self.logPath + os.sep + 'Nip.txt'
20        self.Nip = None
21
22        self.getNip()
23        self.writeNip()
24
25 ####  从网络取得本地的公网 IP
26    def getNip(self):
27        urls = 'http://1111.ip138.com/ic.asp'
28        if urllib2.urlopen(urls).geturl() == urls:
29            rawString = urllib2.urlopen(urls).read()
30            self.Nip = re.search(b'\d+\.\d+\.\d+\.\d+',rawString).group()
31            print("Nip = %s"%self.Nip)
32        else:
33            print("未取得本机 NIP")
34
35 ####  将取得的公网 IP 写入指定文件中
36    def writeNip(self):
37        if os.path.isdir(self.logPath):
38            pass
39        else:
40            os.makedirs(self.logPath)
41        with open(self.nipFile,'w') as FP:
42            FP.write(self.Nip)
43
44
```

```
45 ####  以下是脚本的主程序
46 if __name__ == '__main__':
47     nip = GetNip()
```

测试一下，执行命令得到的结果如图 5-36 所示。

```
python getNip.py
```

```
pi@raspberrypi: ~/code/python/getNip
pi@raspberrypi ~/code/python/getNip $ ls -l
total 4
-rw-r--r-- 1 pi pi 861 Sep  9 20:42 getNip.py
pi@raspberrypi ~/code/python/getNip $ python getNip.py
Nip = 221.235.81.1
pi@raspberrypi ~/code/python/getNip $
```

图 5-36　getNip.py 执行结果

就这样很简单地取得了本地的公网 IP。网络上返回本地 IP 的站点很多。这里只是选择最常用的一个。如果想增强 Script 的健壮性，可以多添加几个站点做后备。


注 意

这种 Script 仅在电信的网络上测试过。如果使用别的网络服务无法取得 IP，没关系。原理清楚了使用哪种方法并不重要。

5.5.2　端口映射

在上节中已经取得了本地的公网 IP。可取得了公网 IP 后，在互联网环境下还是不能访问本地内网的 HTTP 服务器。通常的做法是在网关上做个端口映射，将内网的 HTTP 服务端口映射到网关上。这样就可以通过网关（modem）来访问内网的 HTTP 服务器。

鉴于目前糟糕的网络环境，个人是无法自由地选择 Modem 的，只能使用网络服务商提供的专用 Modem。而网络服务商提供的 Modem 经过一些修改、裁剪，无法进行正常的端口映射（这里以中国电信提供的光猫中兴 F460 为例，下文中未特殊说明的 Modem 都是指中兴 F460）。所以，只有自己动手，才能丰衣足食了。另外，这种方法只适合有公网 IP 的用户。如果连公网 IP 都没有的（比如长城、鹏博士），那还是去用花生壳吧。这种程度的端口映射是不能穿透内网中的内网中的内网中……

1. 原理

电信版中兴 F460 这款光猫应用范围很广，而且自带 WIFI 功能，效果也还不错。可讨厌的地方是无法进行任何设置。Web 网页设置时需要输入超级密码，个人用户是没有这个密码的，即使通过技术手段得到了这个密码，进入了 Web 设置页面，也无法设置端口映射和 DMZ 主机。所以想设置端口映射还得另想办法。

几乎所有的 Modem 都是以嵌入式 Linux（openwrt）为基础改编的。中兴 F460 这款 Modem 当然也不例外。使用 nmap 扫描一下这个光猫。执行命令：

```
nmap -sT -O 192.168.1.1
```

得到的结果如图 5-37 所示

```
king@debian: ~
root@debian:/home/king# nmap -sT -O 192.168.1.1

Starting Nmap 6.47 ( http://nmap.org ) at 2015-09-09 21:07 CST
Nmap scan report for 192.168.1.1
Host is up (0.0051s latency).
Not shown: 996 closed ports
PORT      STATE SERVICE
23/tcp    open  telnet
80/tcp    open  http
5001/tcp  open  commplex-link
52869/tcp open  unknown
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6
OS details: Linux 2.6.16 - 2.6.28
Network Distance: 2 hops

OS detection performed. Please report any incorrect results at http://nmap.org/s
ubmit/ .
Nmap done: 1 IP address (1 host up) scanned in 14.72 seconds
root@debian:/home/king# 
```

图 5-37　nmap modem

可以看到除了 http 端口开放外，还开放了 telnet 端口。既然开放了 telnet 端口，那就说明可以远程登录。如果能登录，那就可以直接以 iptables 来设置端口映射。telnet 的用户名是 root，密码多试几次就出来了，密码还是 root。

现在只需要用 Python 脚本登录 Modem 的 telnet 服务，调用 iptables 将 Raspberry 的端口转发到 Modem 上就可以了。

2. 实际操作

首先要说明的是笔者的拓扑结构，光钎入户连接光猫中兴 F460（192.168.1.1）。光猫的 LAN 口连接 TP-LINK 路由器（192.168.2.1）的 WAN 口。TP-LINK 路由器的 LAN 口连接到 Raspberry（192.168.2.91）和 PC。拓扑图如图 5-38 所示

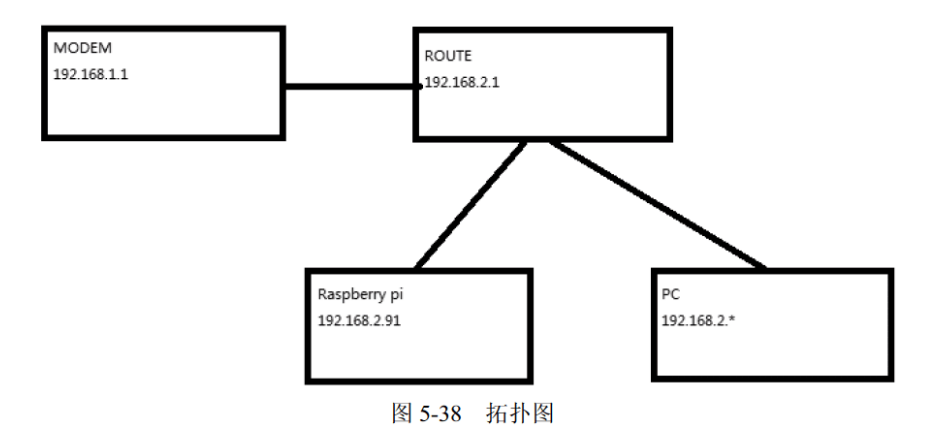


图 5-38　拓扑图

（1）从主机到路由器的映射

先将 Raspberry 的 http 端口映射到 TP-LINK 路由器上。这里我使用的是 DMZ 主机，如图 5-39 所示。



图 5-39　DMZ 主机设置

查看 TP-LINK 在 modem 上的 IP，如图 5-40 所示。



图 5-40　路由器外网 IP

（2）路由器到光猫的映射

编写 portmap.py，将公网 IP 的 8080 端口映射到 TP-LINK 路由器（192.168.1.90）的 8080 端口上，间接地将端口映射到 Raspberry（192.168.2.91）。现在开始来编写 portmap.py，使用 Putty 登录到 Raspberry 上，执行命令：

```
cd
mkdir -pv code/python/portmap
cd $_
touch2py portmap.py
vi portmap.py
```

portmap.py 的代码如下：

```
 1 #!/usr/bin/env python
 2 # -*- coding:utf-8 -*-
 3 #Author   :hstking
 4 #E-mail   :hstking@hotmail.com
 5 #Ctime    :2015/09/09
 6 #Mtime    :
 7 #Version  :
 8
 9 import logging
```

```
10 import os
11 import telnetlib
12 import urllib2
13 import re
14 import time
15
16
17 ####  定义 GetNip 类
18 class GetNip():
19 ####  GetNip 类的构造函数
20     def __init__(self):
21         self.logPath = os.path.expanduser('~') + os.sep + 'log'
22         self.nipFile = self.logPath + os.sep + 'Nip.txt'
23         self.Nip = None
24
25         self.getNip()
26         self.writeNip()
27
28 ####  从网络取得本地的公网 IP
29     def getNip(self):
30         urls = 'http://1111.ip138.com/ic.asp'
31         if urllib2.urlopen(urls).geturl() == urls:
32             rawString = urllib2.urlopen(urls).read()
33             self.Nip = re.search(b'\d+\.\d+\.\d+\.\d+',rawString).group()
34             print("Nip = %s" %self.Nip)
35         else:
36             print("未取得本机 NIP")
37
38 ####  将取得的公网 IP 写入文件
39     def writeNip(self):
40         if os.path.isdir(self.logPath):
41             pass
42         else:
43             os.makedirs(self.logPath)
44         with open(self.nipFile,'w') as FP:
45             FP.write(self.Nip)
46
47
48 ####  定义 PortMap 类
49 class PortMap(object):
50 ####  PortMap 类的构造函数
51     def __init__(self):
52         self.tn = None
53         self.gn = GetNip()
54         self.nip = self.gn.Nip
55         self.ml = MyLog()
56 ####  定义本地的环境变量
57         self.dict1 = {
58                 'modemIp':'192.168.1.1',
```

```
59              'mapIp':'192.168.1.90',
60              'user':b'root',
61              'password':b'root',
62              'finish':b'/ #  '}
63 ####  iptables 命令列表，清除 iptables 环境，以便于之后的设置
64          self.portmap_clear = [
65              'iptables -t nat -F myPREROUTING',
66              'iptables -t nat -D PREROUTING -j myPREROUTING',
67              'iptables -t nat -X myPREROUTING',
68              'iptables -t nat -F myPREROUTING',
69              'iptables -t nat -F myPOSTROUTING',
70              'iptables -t nat -D POSTROUTING -j myPOSTROUTING',
71              'iptables -t nat -X myPOSTROUTING']
72 #### iptables 命令列表，设置 iptables，将 Route 上的 8080 端口映射到 Modem 的 8080 端口上
73          self.portmap_set = [
74              'iptables -t nat -N myPREROUTING',
75              'iptables -t nat -A myPREROUTING -d ' + self.nip + ' -p tcp -m
tcp --dport 8080 -j DNAT --to-destination ' + self.dict1['mapIp'] + ':8080',
76              'iptables -t nat -A PREROUTING -j myPREROUTING',
77              'iptables -t nat -N myPOSTROUTING',
78              'iptables -t nat -A myPOSTROUTING -d ' + self.dict1['mapIp'] +
' -p tcp -m tcp --dport 8080 -j SNAT --to-source ' + self.dict1['modemIp'],
79              'iptables -t nat -A POSTROUTING -j myPOSTROUTING']
80          self.set_iptables()
81
82 ####  该函数用于设置 iptables
83      def set_iptables(self):
84          self.ml.info(u'开始设置 iptables ......')
85          self.conn_telnet()
86          cmd = None
87          for cmd in self.portmap_clear:
88              self.tn.write('%s \n' %cmd)
89              self.ml.info('Run command : "%s" successfull' %cmd)
90              time.sleep(2)
91          self.ml.info(u'iptables 清除完毕')
92          cmd = None
93          for cmd in self.portmap_set:
94              self.tn.write('%s \n' %cmd)
95              self.ml.info('Run command : "%s" successfull' %cmd)
96              time.sleep(2)
97          self.ml.info(u'iptables 设置完毕 ......')
98          self.disconn_telnet()
99
100 ####  该函数用于连接 Modem 上的 telnet 服务
101     def conn_telnet(self):
102         self.tn = telnetlib.Telnet(self.dict1['modemIp'])
103         self.tn.read_until(b'Login: ')
104         self.tn.write(self.dict1['user'] + b'\n')
105         self.tn.read_until(b'Password: ')
```

```
106         self.tn.write (self.dict1['password'] + b'\n')
107         self.tn.read_until (self.dict1['finish'])
108         self.ml.info (u"telnet 连接成功")
109
110 ####  该函数用于断开 Modem 上的 telnet 服务
111     def disconn_telnet (self) :
112         self.tn.close()
113         self.ml.info (u"telnet 断开")
114
115
116 ####  定义一个 MyLog 类
117 class MyLog (object) :
118     def __init__ (self) :
119         self.logger = logging.getLogger ('pi')
120         self.logFile = '/home/pi/log/' + os.path.basename (__file__) [0:-3] +
'.log'
121         self.logger.setLevel (logging.DEBUG)
122         self.formatter = logging.Formatter ('% (asctime) -12s % (levelname) -8s
% (name) -10s % (message) -12s')
123
124         self.logHand = logging.FileHandler (self.logFile)
125         self.logHand.setLevel (logging.DEBUG)
126         self.logHand.setFormatter (self.formatter)
127
128         self.logHandSt = logging.StreamHandler()
129         self.logHandSt.setLevel (logging.DEBUG)
130         self.logHandSt.setFormatter (self.formatter)
131
132
133         self.logger.addHandler (self.logHand)
134         self.logger.addHandler (self.logHandSt)
135
136 ####  这里只定义了一个 info，实际上还可以有 bug,error……
137     def info (self,msg) :
138         self.logger.info (msg)
139
140
141 if __name__ == '__main__':
142     portMap = PortMap()
```

好了，现在执行命令，执行结果如图 5-41 所示。

```
python portmap.py
```

```
pi@raspberrypi ~/code/python/portmap $ python portmap.py
Nip = 221.235.80.64
2015-09-11 00:46:34,283 INFO     pi         开始设置 iptables ......
2015-09-11 00:46:34,380 INFO     pi         telnet 连接成功
2015-09-11 00:46:34,381 INFO     pi         Run command : "iptables -t nat -F my
PREROUTING" successfull
2015-09-11 00:46:36,385 INFO     pi         Run command : "iptables -t nat -D PR
EROUTING -j myPREROUTING" successfull
2015-09-11 00:46:38,389 INFO     pi         Run command : "iptables -t nat -X my
PREROUTING" successfull
2015-09-11 00:46:40,393 INFO     pi         Run command : "iptables -t nat -F my
PREROUTING" successfull
2015-09-11 00:46:42,397 INFO     pi         Run command : "iptables -t nat -F my
POSTROUTING" successfull
2015-09-11 00:46:44,402 INFO     pi         Run command : "iptables -t nat -D PO
STROUTING -j myPOSTROUTING" successfull
2015-09-11 00:46:46,406 INFO     pi         Run command : "iptables -t nat -X my
POSTROUTING" successfull
2015-09-11 00:46:48,410 INFO     pi         iptables 清除完毕
2015-09-11 00:46:48,412 INFO     pi         Run command : "iptables -t nat -N my
PREROUTING" successfull
2015-09-11 00:46:50,416 INFO     pi         Run command : "iptables -t nat -A my
PREROUTING -d 2              -p tcp -m tcp --dport 8080 -j DNAT --to-destination
192.168.1.90:8080" successfull
2015-09-11 00:46:52,421 INFO     pi         Run command : "iptables -t nat -A PR
EROUTING -j myPREROUTING" successfull
2015-09-11 00:46:54,425 INFO     pi         Run command : "iptables -t nat -N my
POSTROUTING" successfull
2015-09-11 00:46:56,429 INFO     pi         Run command : "iptables -t nat -A my
POSTROUTING -d 192.168.1.90 -p tcp -m tcp --dport 8080 -j SNAT --to-source 192.1
68.1.1" successfull
2015-09-11 00:46:58,433 INFO     pi         Run command : "iptables -t nat -A PO
STROUTING -j myPOSTROUTING" successfull
2015-09-11 00:47:00,437 INFO     pi         iptables 设置完毕 ......
2015-09-11 00:47:00,440 INFO     pi         telnet 断开
pi@raspberrypi ~/code/python/portmap $
```

图 5-41　端口映射

（3）映射验证

最后来验证一下，如图 5-42 所示。

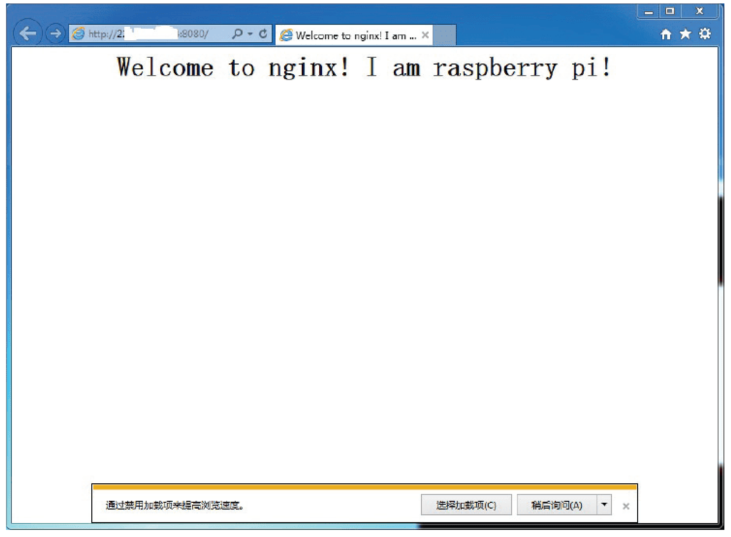


图 5-42　公网访问内网

好了，现在无须花生壳也可以从公网访问内网的服务了。用同样的方法也可以将 FTP 服务、ssh 服务……将服务器端口都映射到光猫上。



在 Linux 中的端口映射，实质上就是 iptables 的转发。

5.6 实战：Raspberry 给自己发短信

上节已经将内网的 http 服务端口映射到公网上了。现在的问题是，如果身处内网没必要知道公网的 IP，可以直接通过内网 IP 访问 http 服务。如果身处外网又无法执行 getNip.py 得到外网的 IP，似乎陷入死循环了。我的解决方案是通过 sendMess.py 给自己的手机发送外网的 IP，这样就可以通过内网的公网 IP 来访问内网的 http 服务。

5.6.1 方案原理

怎样才能让 Python 给自己的手机发短信呢？方案有两个，第一是通过飞信的接口给自己发短信。可这种方案极不稳定，说不定哪天飞信的接口就被封了。那就只有选择第二种方案了，给特定的邮箱发邮件。邮箱会将邮件内容发送到绑定的手机上。目前这种邮箱很多，如 163.mail，139.mail……我选择的是 139 邮箱。如果实在是找不到免费手机提醒功能的邮箱，那就设置成 QQ 邮件吧，QQ 邮箱收到邮件后会通知微信，这样也能凑合。这个脚本的本质上就是自动发送邮件。



使用 Raspberry 直接发短信也是可以的，那需要添加其他的模块。

5.6.2 方案执行

可以将上节的脚本都用上。首先是提取 Nip.txt 中保存的 nip，然后利用 getNip.py 获取当前公网 IP。比较获取的公网 IP 和提取保存的 nip，相同则什么都不需要做，不同则将获取的公网 IP 发送到邮箱。使用 Putty 登录 Raspberry，执行命令：

```
cd
mkdir -pv code/python/sendMsg
cd $_
touch2py sendMsg.py
cp ../getNip/getNip.py ./
vi sendMsg.py
```

sengMsg.py 的代码如下：

```
1 #!/usr/bin/env python
2 # -*- coding:utf-8 -*-
3 #Author  :hstking
4 #E-mail  :hstking@hotmail.com
5 #Ctime   :2015/09/11
6 #Mtime   :
```

```
 7 #Version :
 8
 9 import smtplib
10 import email.utils
11 from email.mime.text import MIMEText
12 import getNip
13
14 ####  定义 SendMail 类
15 class SendMail (object) :
16 ####  定义 SendMail 的构造函数
17     def __init__ (self,subject,content) :
18         self.subject = subject
19         self.content = content
20 ####  self.mailList 是接收邮件的地址列表
21         self.mailList = ['139********@139.com','33*******@qq.com']
22 ####  self.fromMail 是发送邮件的地址
23         self.fromMail = 'hst****@163.com'
24 ####  self.mi 是发送邮件的用户名密码
25         self.mi = {'user':'hst****','password':'*************'}
26
27         self.sendMail (self.subject,self.content)
28
29 ####  sendMail 函数用于发送邮件
30     def sendMail (self,subject,content) :
31         for mL in self.mailList:
32             msg = MIMEText (content)
33             msg['To'] = email.utils.formataddr ( (mL[:mL.index ('@') ],mL) )
34                                    msg['From']   =   email.utils.formataddr
    ( (self.fromMail[:self.fromMail.index ('@') ],self.fromMail) )
35             msg['Subject'] = subject
36             try:
37                 s = smtplib.SMTP()
38                 s.set_debuglevel (1)
39                 s.connect ('smtp.' + self.fromMail[self.fromMail.index ('@')  +
1:])
40                 s.login (self.mi['user'],self.mi['password'])
41                 s.sendmail (self.fromMail,mL,msg.as_string())
42             except EOFError,e:
43                 print str (e)
44             finally:
45                 s.quit
46
47
48 ####  定义 SendMsg 类
49 class SendMsg (object) :
50 ####  定义 SendMsg 的构造函数
51 ####  通过 getNip.GetNip () 函数取得本地的公网 IP，将取得的公网 Ip 与之前保存的 Ip 比较
52 ####  如果保存文件不存在或保存的 IP 与取得的 Ip 不同，则发送邮件
53     def __init__ (self) :
```

```
54        nipFile = '/home/pi/log/Nip.txt'
55        with open (nipFile,'r') as fp:
56            self.nip = fp.read()
57        Nip = getNip.GetNip()
58        if  self.nip == Nip.Nip:
59            pass
60        else:
61            sMsg = SendMail ('IP',self.nip)
62
63
64 if __name__ == '__main__':
65     SMSG = SendMsg()
```

好了，现在可以使用这个脚本向手机上发送私人服务器的公网 IP 了。这个脚本中，import 不光载入 Python 的标准模块，还使用 import getNip 将 getNip.py 当成模块载入脚本中使用。使代码能够重复利用，避免重复地造轮子。当前 Raspberry 默认的编辑器是 nano，先将它改成 vi。执行命令：

```
export EDITOR=vi
crontab -e
```

开始编辑例行性任务 crontab，crontab 里面以#开头的都是注释，可以全部删除。在最后一行添加：

```
5 * * * * python /home/pi/code/python/sendMsg/sendMsg.py
```

保存退出，使用 crontab –l 检查结果，如图 5-43 所示。

```
pi@raspberrypi: ~/code/python/sendMsg
pi@raspberrypi ~/code/python/sendMsg $ crontab -l
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
5 * * * * python /home/pi/code/python/sendMsg/sendMsg.py
pi@raspberrypi ~/code/python/sendMsg $
```

图 5-43　例行性任务

crontab 中，第一个 5 是指每 5 分钟，第二个*是指每小时，第三个*是指每天，第四个*是指每月，第五个*是指每周。这样就设置了一个例行性任务，每 5 分钟获取一次 Nip，将得到的 Nip 和之前保存的 Nip 比较。相同则什么都不做，如果不同，就发送邮件到指定的邮箱。然后由邮箱发送短信到手机上。


注 意

crontab 是 Linux 例行性命令。

5.7 监控器 Motion

Motion 是一个相当轻量级，但却能够在 Linux 上运行监控摄像头的应用。它可以和任何支持 Linux 的摄像头一起工作，包括所有 V4L（Video4Linux，Linux 内核中关于视频设备的 API 接口）网络摄像头、许多 IP 网络摄像头和 Axis 摄像头。Motion 还能够控制云台功能。Motion 以 JPEG、PPM 和 MPEG 格式存储影像和快照。由于 Motion 内置了 http 服务器，我们可以在网络浏览器中进行远程观看。虽然 Motion 支持 MySQL 和 PostgreSQL 数据库，但是它仍然可以在不需要数据库的情况下，将图片文件存储在你选择的目录。

在 Debian 及其衍生版本上安装 Motion 非常容易，因为它本身已经包含了所有必需的软件库。因此我们所需要的仅仅是运行 apt-get install motion，还需要 libav-tools，它是一个 ffmpeg 分支。Debian 用 libav-tools 取代了 ffmpeg。

5.7.1 安装 Motion

RaspBian 的官方源里包含了 Motion，只需要执行命令：

```
sudo apt-get install motion
sudo apt-get install libav-tools
```

5.7.2 配置使用 Motion

将摄像头插入 Raspberry 的 USB 接口。用 Putty 登录 Raspberry。执行命令：

```
lsusb
```

查看摄像头是否能被 Raspberry 识别。如果能被自动识别，结果如图 5-44 所示。

```
pi@raspberrypi: ~
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 18ec:3399 Arkmicro Technologies Inc.
pi@raspberrypi ~ $
```

图 5-44 显示设备

上图中 Bus 001 Device 004 就是刚插入的 USB 摄像头。如果不能被识别，就去找合适的驱动安装驱动，或者查看摄像头芯片后编译内核，将合适的驱动编入内核或模块。

Motion 的配置文件在/etc/motion/motion.conf，下面开始修改 Motion 的配置文件。登录 Raspberry，执行命令：

```
cd /etc/motion
sudo cp motion.conf motion.conf.bak
sudo sed -i 's/^daemon on/daemon off/g' motion.conf
sudo sed -i 's/no/yes/g' /etc/default/motion
sudo sed -i 's/webcam_localhost on/webcam_localhost off/g' motion.conf
```

第一个和第二个 sed 修改的是确认 Motion 以后台模式运行。第三个 sed 修改的是解除 Motion 只能在本机下查看的限制。

启动 Motion 服务，执行命令：

```
sudo /etc/init.d/motion restart
```

好了，现在 Motion 已经可以使用了。Motion 默认使用 8081 端口，用 nmap 查看一下，如图 5-45 所示.

```
pi@raspberrypi: /etc/default
pi@raspberrypi /etc/default $ nmap 192.168.2.91 -p 8081

Starting Nmap 6.00 ( http://nmap.org ) at 2015-09-19 01:07 CST
Nmap scan report for 192.168.2.91
Host is up (0.00034s latency).
PORT      STATE SERVICE
8081/tcp open  blackice-icecap

Nmap done: 1 IP address (1 host up) scanned in 0.49 seconds
pi@raspberrypi /etc/default $
```

图 5-45　扫描确认端口

打开 Firefox 浏览器，在地址栏输入 http://192.168.2.91:8081。不要使用 Chromium 和 IE。Chromium 不支持，IE 效果不太好。在 Raspberry 上连接一个网络摄像头，然后将 8081 端口也映射到公网上，就可以将它当成一个远程监控器来使用。


注　意

Raspberry 使用远程摄像头也有其他的选择，但 Motion 是最简单的，也是可扩展性最好的。

第 6 章

◂ 实战Raspberry GPIO ▸

Raspberry 可以通过扩展控制其他的电子元件、模块。那 Raspberry 是怎么控制电子元件、模块的呢？这里就不得不说道 GPIO 了。本章通过学习 GPIO 了解简单的电子模块。为复杂的应用做基本的技术储备。

本章主要内容包括：

- 认识 GPIO
- 控制 LED 二极管闪烁
- 驱动蜂鸣器
- 驱动超声波模块

6.1 GPIO 简介

General Purpose Input Output（通用输入/输出）简称为 GPIO，或总线扩展器，利用工业标准 I2C、SMBus 或 SPI 接口简化了 I/O 口的扩展。当微控制器或芯片组没有足够的 I/O 端口，或当系统需要采用远端串行通信或控制时，GPIO 产品能够提供额外的控制和监视功能。

6.1.1 Raspberry GPIO

Raspberry2 采用的是 40pin 的 GPIO，GPIO 的编号方法有些混乱，不同的 API（如 wiringPi，RPi.GPIO 等）对 GPIO 的端口号编号并不一样。

- wiringPi: 有 Perl、PHP、Ruby、Node.JS 和 Golang 的扩展，支持 wiringPi Pin 和 BCM GPIO 两种编号。
- RPi.GPIO：Python，支持 Board Pin 和 BCM GPIO 两种编号。
- Webiopi：Python，使用 BCM GPIO 编号。
- WiringPi-Go：Go 语言，支持以上三种编号。

这里采用了官方推荐的 RPi.GPIO，也顺便选择了资料最丰富的 BCM GPIO 编号。GPIO 端口编号如图 6-1 所示。

```
+-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
|     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
|   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
|   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
|   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
|     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
|  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
|  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
|  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
|     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
|  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
|   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
|  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
|     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
|   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
|   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
|   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
|  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
|  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
|  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
|     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
+-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
| BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
+-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
```

图 6-1　GPIO 编号

BCM2835 编号方式侧重 CPU 寄存器。 wiringPi 编号方式侧重实现逻辑，把扩展 GPIO 端口从 0 开始编号，这种编号方便编程。

6.1.2　物理端口

Raspberry GPIO 物理端口顺序是从上到下，从左到右。再来看看这些端口的用途。

这些端口大致可以分成三类。

第一类是电源物理端口，看 Name 项中，它被标注为 3.3v、0v、5v 的，要么就是直流电源的火线，要么就是 groud 地线。

第二类是 GPIO 的控制端口，看 Name 项中，名字是 GPIO.*的都是 GPIO 的控制端口，我们就是通过 python 控制这些端口的高电平，低电平来控制使用模块功能的。

第三类是剩下的那些端口了。如 Name 项中标注成 RxD、TxD、SDA、SCL……这些端口稍微高端一点，RxD、TxD 是 Receive Data、Transmit Data 的意思。RxD 为接收数据的引脚，TxD 为发送数据的引脚。SDA 是双向数据线，SCL 是时钟线。在 I2C 总线上传送数据，首先送最高位，由主机发出启动信号，SDA 在 SCL 高电平期间由高电平跳变为低电平，然后由主机发送一个字节的数据。数据传送完毕，由主机发出停止信号，SDA 在 SCL 高电平期间由低电平跳变为高电平。目前暂时用不上，就不在此详细介绍了。

6.2　实战 GPIO——LED 呼吸灯

控制 GPIO 端口，可以用 C、Python、Go……个人认为 Python 是最简单的，而且 Python 支持多平台，可以在不同的系统下执行。这是别的语言无法比拟的。所以这里选择用 Python 来控制 GPIO 端口。本节将使用 Python 编程，通过 GPIO 端口来控制一盏 LED 二极管闪烁。初步了解 Raspberry

从软件到硬件的过程。这是一个最简单的 GPIO 实验。

6.2.1 准备实验物品

既然是 Raspberry 控制 LED 二极管，Raspberry 和 LED 二极管那是必需的了，LED 发光二极管的颜色任选，如图 6-2 所示。



图 6-2 LED 二极管

Raspberry pi2 扩展板，在上面有 BCM 的端口编号，还是很方便的，如图 6-3 所示。不必须，但是如果有，肯定会方便很多。

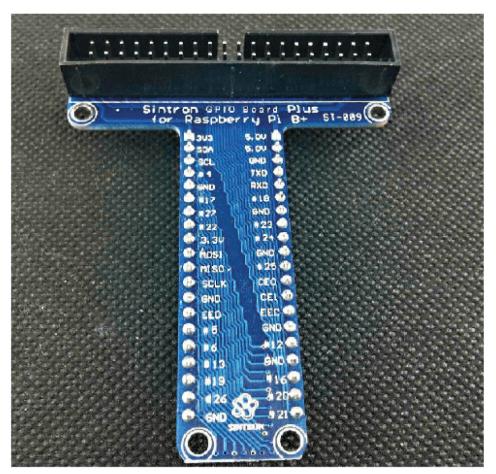


图 6-3 Raspberry pi2 扩展板

40pin 的排线，如果不想频繁地开 Raspberry 的盖子，它还是值得拥有的，如图 6-4 所示。

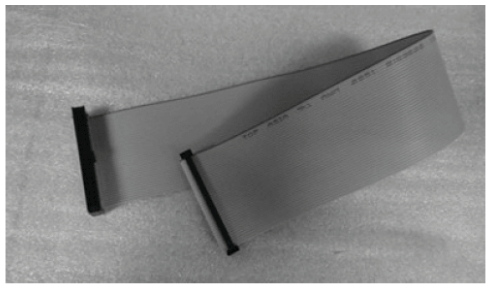


图 6-4 40pin 排线

面包板，电子实验必备工具。在这里要是没有也行，无非就是麻烦点，如图 6-5 所示。



图 6-5　面包板

好了，把这些实验用品按照图 6-6 所示安装好。

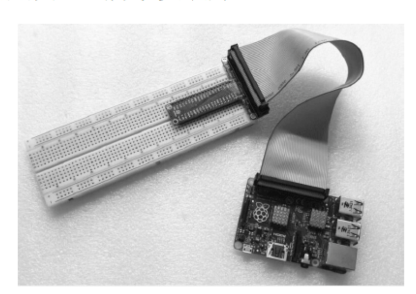


图 6-6　组装设备

LED 二极管的管脚一长一短。长管脚接正极，短管脚接负极。这里是准备将长管脚接入扩展板上标有数字的端口，短管脚接入 GND 端口。所以这里可选择 5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27 这几个端口。因为 LED 二极管的短脚要接入 GND 端口，所以尽量选择 GND 端口相邻的控制端口。这里我选择的是扩展板上标注为 12 的控制端口，如图 6-7 所示。

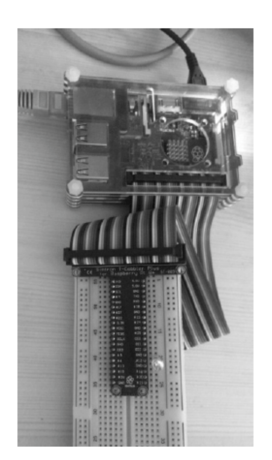


图 6-7　LED 呼吸灯实验

硬件的准备工作已经完毕了，下面只需要专心 Python 控制就可以了。

6.2.2 Python 控制

这里选用的是 RPi.GPIO 编程，好在 RaspBian 已经默认安装好了 RPi.GPIO。不用费心思去下载，直接使用就可以了。这个实验的原理很简单，就是通过编程来控制某个端口的电平。当高电平时，LED 二极管就亮了，低电平时二极管就熄了。由此推彼，用 Raspberry 的 GPIO 连接其他的设备，也只是控制电平的高低而已。执行命令：

```
cd
mkdir -pv code/python/light
cd code/python/llight
touch2py light.py
vi light.py
```

light.py 代码如下：

```
 1 #!/usr/bin/env python
 2 # -*- coding:utf-8 -*-
 3 #Author  :hstking
 4 #E-mail  :hstking@hotmail.com
 5 #Ctime   :2015/08/30
 6 #Mtime   :
 7 #Version :
 8
 9 import RPi.GPIO as GPIO
10 import time
11 import sys
12 import string
13
14
15 ####  定义 Light 类
16 class Light (object) :
17 ####  定义 Light 类的构造函数
18    def __init__ (self,pin) :
19        self.pin = pin
20 ####  self.pins 是可以使用的端口列表
21        self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]
22        self.up_time = 0.5
23        self.down_time = 0.5
24        self.check_pin (pin)
25        self.run ()
26
27    def run (self) :
28        self.setup (self.pin)
29        try:
30            self.loop (self.pin)
31        except KeyboardInterrupt:
32            self.destroy (self.pin)
```

```
33
34 ####  check_pin 函数负责检测输入的端口是否符合要求
35    def check_pin (self,pin) :
36        if pin in self.pins:
37            print ("%d 是有效编号"%pin)
38        else:
39            print ("只能输入以下有效的 pin 编号")
40            for i in self.pins:
41                print i,
42            exit()
43
44 ####  setup 函数将端口初始化
45    def setup (self,pin) :
46        #初始化 GPIO 口
47        #采用 BCM 编号
48        GPIO.setmode (GPIO.BCM)
49        #设置 GPIO 为输出状态，输入低电平
50        GPIO.setup (pin,GPIO.OUT)
51        GPIO.output (pin,GPIO.LOW)
52
53 ####  loop 函数将 LED 灯循环点亮
54    def loop (self,pin) :
55        for i in xrange (1,10) :
56            GPIO.output (pin,GPIO.HIGH)
57            print ("light up")
58            time.sleep (self.up_time)
59            GPIO.output (pin,GPIO.LOW)
60            print ("light down")
61            time.sleep (self.down_time)
62
63
64 ####  恢复 GPIO 口状态
65    def destroy (self,pin) :
66        GPIO.output (pin,GPIO.LOW)
67        GPIO.setup (pin,GPIO.IN)
68
69 if __name__ == '__main__':
70    light = Light (12)
```

好了，软硬件都准备好了。直接在 light.py 目录下执行命令 sudo python light.py 就可以看到结果了。


注 意

这个实验是 GPIO 最简单的应用。先把这个实验弄懂，后面的实验才会事半功倍。

6.3　实战 GPIO——蜂鸣器

从上节的实验中可以大致了解 Raspberry 通过控制 GPIO 端口间接地控制其他电子原件、模块的过程和原理。本节的目的是初步了解蜂鸣器模块，使用 Python 编程驱动蜂鸣器，为后面的 Raspberry 报警器做前期技术储备。

6.3.1　准备实验物品

这次的实验物品与 LED 二极管差不多，只是多了一个有源蜂鸣器模块，如图 6-8 所示。



图 6-8　蜂鸣器模块

该模块采用 S8050 三极管驱动，工作电压 3.3V-5V，当 I/O 口输入高电平时，蜂鸣器发声。VCC 针脚外接 3.3V-5V 电压，GND 针脚外接 GND，I/O 针脚外接 Raspberry 的 gpio 控制端口。

其次需要的是杜邦线，这个是做一般电子实验必须的物品。蜂鸣器模块的 3 个针脚这么近，不太可能在面包板上找到合适的位置，所以使用杜邦线是唯一的选择了。杜邦线的接头选择公对母的，如图 6-9 所示。



图 6-9　杜邦线

所有实验物品到位后，按照图 6-10 安装起来，在这里我选择的 GPIO 控制端口是 18 号端口。

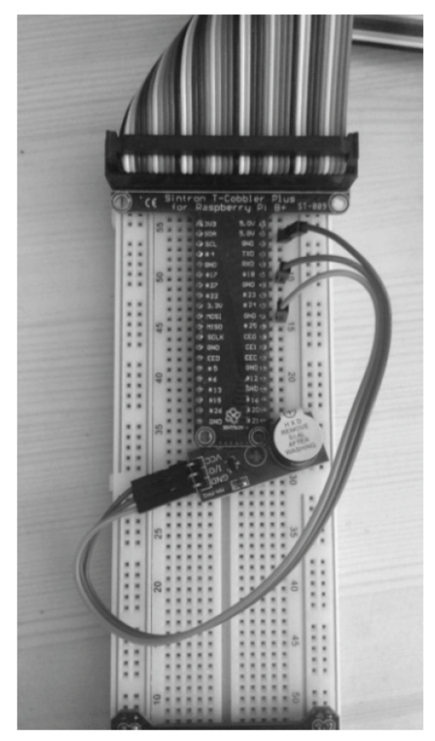


图 6-10　蜂鸣器模块实验

硬件准备工作完毕，下面准备 Python 控制。

6.3.2　Python 控制

实际上，蜂鸣器的控制脚本跟 LED 二极管的控制脚本非常相似。或者说，几乎所有的模块都很相似。因为都是用 Python 来控制某个端口的高低电平。不同的只是高低电平的时间和控制的端口号而已。使用 Putty 登录 Raspberry 后，执行命令：

```
cd
mkdir -pv code/Python/bell
cd code/Python/bell
touch2py bell.py
vi bell.py
```

bell.py 的代码如下：

```
1 #!/usr/bin/env Python
2 # -*- coding:utf-8 -*-
3 #Author  :hstking
4 #E-mail  :hstking@hotmail.com
5 #Ctime   :2015/08/30
6 #Mtime   :
7 #Version :
8
```

```
 9 import RPi.GPIO as GPIO
10 import time
11 import sys
12 import string
13
14
15 ####  定义 Bell 类
16 class Bell (object) :
17 ####  定义 Bell 类的构造函数
18     def __init__ (self,pin) :
19         self.pin = pin
20         self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]
21         self.up_time = 1.5
22         self.down_time = 0.5
23         self.check_pin (pin)
24         self.run()
25
26     def run (self) :
27         self.setup (self.pin)
28         try:
29             self.loop (self.pin)
30         except KeyboardInterrupt:
31             self.destroy (self.pin)
32
33 ####  检查端口是否符合要求
34     def check_pin (self,pin) :
35         if pin in self.pins:
36             print ("%d 是有效编号"%pin)
37         else:
38             print ("只能输入以下有效的 pin 编号")
39             for i in self.pins:
40                 print i,
41             exit()
42
43 ####  初始化端口
44     def setup (self,pin) :
45         #采用 BCM 编号
46         GPIO.setmode (GPIO.BCM)
47         #设置 GPIO 为输出状态，输入低电平
48         GPIO.setup (pin,GPIO.OUT)
49         GPIO.output (pin,GPIO.LOW)
50
51 ####  循环给端口输出高电平
52     def loop (self,pin) :
53         for i in xrange (1,10) :
54             GPIO.output (pin,GPIO.HIGH)
55             print ("bell up")
56             time.sleep (self.up_time)
57             GPIO.output (pin,GPIO.LOW)
```

```
58              print("bell down")
59              time.sleep(self.down_time)
60
61      def destroy(self,pin):
62          #恢复 GPIO 口状态
63          GPIO.output(pin,GPIO.LOW)
64          GPIO.setup(pin,GPIO.IN)
65
66 if __name__ == '__main__':
67      bell = Bell(18)
```

好了，测试一下。直接在 bell 目录下执行命令 sudo Python bell.py 就可以看到结果了。


注 意

这个实验比呼吸灯的实验仅多出了一个时间上的控制，几乎是没什么区别。

6.4 实战 GPIO——超声波模块

蜂鸣器模块只需要发出高电平信号即可。而超声波模块不断要发出高电平信号，而且还要检测收到的信号，仅比蜂鸣器模块稍微复杂一点点。本节的目的是初步了解超声波模块，使用 Python 编程驱动超声波模块，通过超声波模块进行测距。

6.4.1 准备实验物品

本次的实验物品与 LED 二极管的差不多，多了一个超声波模块，如图 6-11 所示。

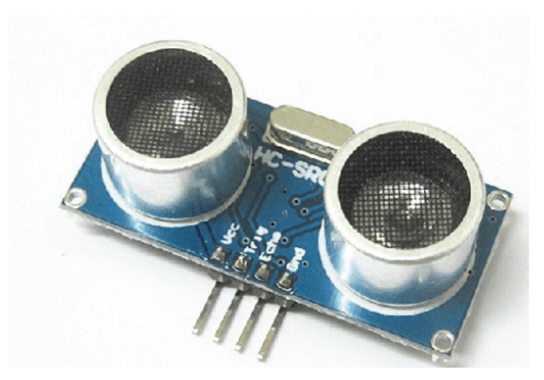


图 6-11 超声波模块

超声波模块给脉冲触发引脚（trig）输入一个长为 20us 的高电平方波。输入方波后，模块会自动发射 8 个 40kHz 的声波。与此同时，回波引脚（echo）端的电平会由 0 变为 1（此时应该启动定时器计时）。当超声波返回被模块接收时，回波引脚端的电平会由 1 变为 0（此时应该停止定时器计数）。定时器记下的这个时间即为超声波由发射到返回的总时长。根据声音在空气中的速度为 344 米/秒， 即可计算出所测的距离。测试距离=（高电平时间*声速（340m/s））/2。

该模块使用电压 DC5V，静态电流小于 2mA，高电平输出 5V，低电平输出 0V，探测距离

2cm~450cm。共有 4 个针脚，VCC 针脚外接 5V 电压；GND 针脚外接 GND；Trig 针脚外接 GPIO 的控制端口，作为控制端口；echo 针脚外接 GPIO 控制端口，作为接收信号端口。

这里我选择的是 GPIO 控制端口的 23 号端口连接 trip 引脚，24 号端口连接 echo 引脚，如图 6-12 模块。

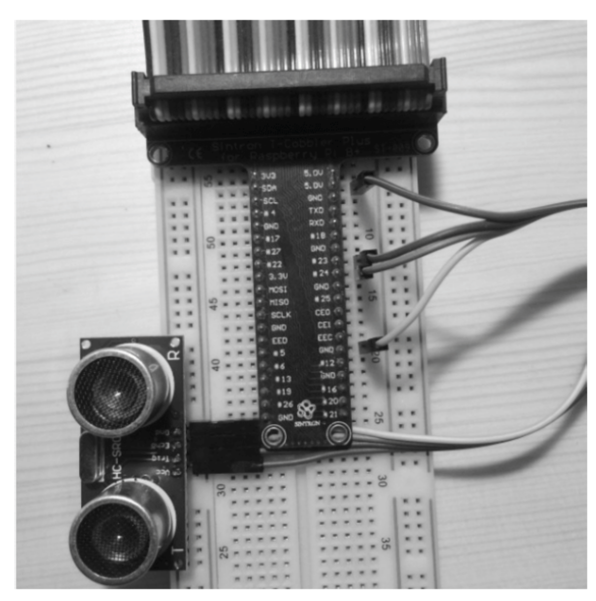


图 6-12　超声波模块实验

硬件组装完毕，下面准备 Python 控制。

6.4.2　Python 控制

超声波模块与 LED 二极管、蜂鸣器模块区别不大。只是针脚端口多一点点，也稍微复杂一点点，再就是最后多出了一个测距。使用 Putty 登录 Raspberry 后，执行命令：

```
cd
mkdir -pv code/Python/ultrasonic
cd code/Python/ ultrasonic
touch2py ultrasonic.py
vi ultrasonic.py
```

ultrasonic.py 的代码如下：

```
1 #!/usr/bin/env Python
2 # -*- coding:utf-8 -*-
3 #Author   :hstking
4 #E-mail   :hstking@hotmail.com
5 #Ctime    :2015/08/30
6 #Mtime    :
7 #Version  :
8
```

```
 9 import RPi.GPIO as GPIO
10 import time
11
12
13 ####  定义类Ultrasonic
14 class Ultrasonic(object):
15 ####  类Ultrasonic的构造函数
16     def __init__(self,trig_pin,echo_pin):
17         self.trig_pin = trig_pin
18         self.echo_pin = echo_pin
19         self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]
20         self.check_pin(self.trig_pin,self.echo_pin)
21         self.run(self.trig_pin,self.echo_pin)
22
23 ####  check_pin函数检测输入的端口是否可用
24     def check_pin(self,trig_pin,echo_pin):
25         if trig_pin in self.pins:
26             print("%d 是有效编号"%trig_pin)
27         else:
28             print("只能输入以下有效的pin编号")
29             for i in self.pins:
30                 print i,
31             exit()
32         if echo_pin in self.pins:
33             print("%d 是有效编号"%echo_pin)
34         else:
35             print("只能输入以下有效的pin编号")
36             for i in self.pins:
37                 print i,
38             exit()
39
40 ####  setup函数初始化GPIO口
41     def setup(self,trig_pin,echo_pin):
42         #采用BCM编号
43         GPIO.setmode(GPIO.BCM)
44         GPIO.setwarnings(False)
45         #设置GPIO为输出状态，输入低电平
46         GPIO.setup(self.trig_pin,GPIO.OUT,initial=GPIO.LOW)
47         GPIO.setup(self.echo_pin,GPIO.IN)
48
49 ####  run函数测距
50     def run(self,trig_pin,echo_pin):
51         self.setup(self.trig_pin,self.echo_pin)
52         #发出触发信号
53         GPIO.output(self.trig_pin,GPIO.HIGH)
54         #保持10us以上
55         time.sleep(0.000015)
56         GPIO.output(self.trig_pin,GPIO.LOW)
57         while not GPIO.input(self.echo_pin):
```

```
58              pass
59          #echo端口发现高电平，开始计时
60          t1 = time.time()
61          while GPIO.input(self.echo_pin):
62              pass
63          #echo端口高电平停止，结束计时
64          t2 = time.time()
65          length = (t2-t1)*340/2
66          print("测试距离为 %0.2f m"%length)
67          return length
68
69
70 if __name__ == '__main__':
71     ul = Ultrasonic(23,24)
```

好了，现在可以通过这个模块进行测距了。超声波模块测距精度比较高，但距离很短。只适合用在较特殊的场合。


注 意

超声波模块实验稍微复杂一点点，明白了原理也很容易理解。这些都是比较简单的模块，它们仅涉及了高低电平的变化，没有涉及寄存器数据。

第 7 章
实战：智能开门报警器

第 6 章已经熟悉了几个模块了，单独的模块并没有什么太大的用处，但组合起来用处就比较多了。本节使用 Raspberry 和以上的模块组合起来打造一个开门报警器，作用是开启门的时候发出报警。

本章主要内容包括：

- 了解报警器需要的硬件
- 了解报警器组装原理
- 了解开发报警器需要的软件
- 实现一个智能报警器

7.1 硬件准备

既然是报警器，那可以通过声音报警和灯光报警，触发器可以是红外也可以选择超声波模块。所需的模块就是上章已经学习过的模块。本节将这几个模块组合起来学以致用，制作一个开门报警器。这个报警器使用的模块不多，功能简单。如果需要添加其他的功能，可以自行扩展。心有多大，天地就有多宽。用在这里似乎也很合适。

7.1.1 必需的硬件

Raspberry 及标准配件一套，这是必不可少的。

蜂鸣器模块一个：既然是报警，蜂鸣器是必不可少的。

Led 二极管一个：有了声音报警，当然还得有报警灯了。

超声波模块一个：把上几节使用的模块一网打尽。

杜邦线：这个当然是必不可少的了。

7.1.2 可选硬件

红外模块：实际上用红外模块做报警器是最好的选择。

面包板：Raspberry pi2 扩展板。有了它们会轻松很多。

摄像头模块：有了它甚至可以拍照留底。

移动电源一套。

7.1.3 组装及原理

将 Raspberry 及标准配件组装好，连接上面包板、开发板，如图 7-1 所示。

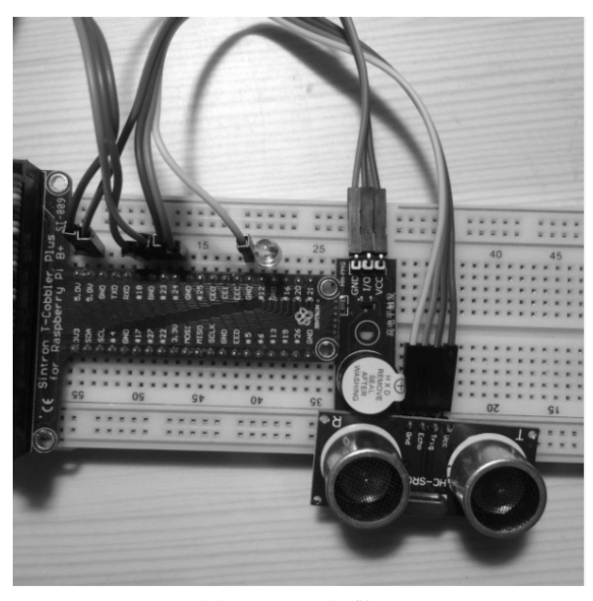


图 7-1 开门报警器

这里二极管选择的是 GPIO 的 12 号端口，超声波模块选择的是 23、24 号端口，蜂鸣器选择的是 18 号端口。

这个开门报警器主要是利用 HC-SR04 超声波模块。HC-SR04 超声波测距模块的精准探测距离是 2cm~450cm。在这个距离之内能保持最大的精度，超出 450cm 后仍能测出距离变化，但测量精度无法保证了。实际应用中我探测到的最大距离是 1400cm。使用 HC-SR04 超声波模块进行测距。如果测出的距离是一个事先固定的距离则说明门没开，如果测出的距离变大，则说明门被打开了，就可以使用蜂鸣器发出警报，Led 二极管开始闪烁，并自动给手机发出消息。

其实报警器使用红外模块的效果可能会更好，如果加上了摄像头模块，还可以将开门的人摄像并发送到手机。

7.2 软件准备

Python 脚本的优点之一是可以非常方便地重利用代码。将原有的代码载入重利用，无须重复造轮子。代码的重利用有两种方法：

第一种是将需要重用的代码直接拷贝到当前目录下当模块执行，如 5.6.2 中的重载 getNip.py。

第二种是在需要重载的代码目录下添加__init__.py，将其模块化。在需要该模块的脚本中执行 sys.path.append 命令，将其目录添加到 sys.path 列表中。

7.2.1 创建 mylog 模块

在 code 文件夹下创建 mylog/mylog.py，然后将其模块化导入到脚本中。演示第二种导入模块的方法。实际上很简单，就是在 mylog 文件夹下创建一个空的__init__.py 文件。这样 Python 会认为这个文件夹是个包。使用 Putty 登录到 Rasypberry，执行命令：

```
cd
cd code/Python
mkdir -pv mylog
cd $_
sudo mkdir -pv /root/log/
touch2py __init__.py
touch2py mylog.py
vi mylog.py
```

以下是 mylog.py 的代码：

```
 1 #!/usr/bin/env Python
 2 # -*- coding:utf-8 -*-
 3 #Author  :hstking
 4 #E-mail  :hstking@hotmail.com
 5 #Ctime   :2015/09/15
 6 #Mtime   :
 7 #Version :
 8
 9 import logging
10 import getpass
11 import os
12 import sys
13
14
15 #### 定义 MyLog 类
16 class MyLog(object):
17 #### 类 MyLog 的构造函数
18     def __init__(self):
19         self.user = getpass.getuser()
20         self.logger = logging.getLogger(self.user)
21         self.logger.setLevel(logging.DEBUG)
22 ####  日志目录
23         self.logPath = '/home/pi/log'
24 ####  日志文件名
25         self.logFile = self.logPath + os.sep + sys.argv[0][0:-3] + '.log'
26         self.formatter = logging.Formatter('%(asctime)-12s %(levelname)-8s
%(name)-10s %(message)-12s')
27
28 ####  日志显示到屏幕上并输出到日志文件内
29         self.logHand = logging.FileHandler(self.logFile)
30         self.logHand.setFormatter(self.formatter)
31
32         self.logHandSt = logging.StreamHandler()
```

```
33         self.logHandSt.setFormatter (self.formatter)
34
35         self.logger.addHandler (self.logHand)
36         self.logger.addHandler (self.logHandSt)
37
38 ####  日志的 5 个级别对应以下的 5 个函数
39     def debug (self,msg) :
40         self.logger.debug (msg)
41
42     def info (self,msg) :
43         self.logger.info (msg)
44
45     def warn (self,msg) :
46         self.logger.warn (msg)
47
48     def error (self,msg) :
49         self.logger.error (msg)
50
51     def critical (self,msg) :
52         self.logger.critical (msg)
53
54 if __name__ == '__main__':
55     mylog = MyLog()
56     mylog.debug ("I'm debug")
57     mylog.info ("I'm info")
58     mylog.warn ("I'm warn")
59     mylog.error ("I'm error")
60     mylog.critical ("I'm critical")
```

好了，现在已经将 mylog 模块化了。只需要使用 sys.path.append 命令将其加入模块路径中就可以直接使用了。


注 意

使用__init__.py 将文件夹模块化，再到需要导入的该模块的脚本中使用 sys.path.append 将该模块的路径导入模块路径。私人定制的模块，就是这么简单。

7.2.2 Python 控制

先将所需的脚本拷贝到当前目录下，以便于代码重用。也就是第一种方法的代码重用。再来构建报警器的主程序。执行命令：

```
cd
cd code/Python
mkdir -pv alarm
cd $_
cp ../sendMsg/* ./
touch2py alarm.py
vi alarm.py
```

以下是 alarm.py 的代码：

```
1 #!/usr/bin/env Python
2 # -*- coding:utf-8 -*-
3 #Author  :hstking
4 #E-mail  :hstking@hotmail.com
5 #Ctime   :2015/09/15
6 #Mtime   :
7 #Version :
8
9 import RPi.GPIO as GPIO
10 import time
11 import sys
12 import string
13 mylogPath = '/home/pi/code/Python/mylog'
14 if not mylogPath in sys.path:
15     sys.path.append(mylogPath)
16 import mylog
17 ####  mylog 模块是通过 mylog 目录下的__init__.py 将目录模块化后导入的
18 import sendMsg
19 ####  sendMsg 模块是将 sendMsg.py 直接拷贝到本地文件夹下直接导入的
20
21
22 ####  定义 Bell 类
23 class Bell(object):
24 ####  Bell 类的构造函数
25     def __init__(self,pin):
26         self.pin = pin
27         self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]
28         self.up_time = 1.5
29         self.down_time = 0.5
30         self.check_pin(pin)
31         self.run()
32
33     def run(self):
34         self.setup(self.pin)
35         try:
36             self.loop(self.pin)
37         except KeyboardInterrupt:
38             self.destroy(self.pin)
39
40 ####  检查输入的端口是否合法
41     def check_pin(self,pin):
42         if pin in self.pins:
43             print("%d 是有效编号"%pin)
44         else:
45             print("只能输入以下有效的 pin 编号")
46             for i in self.pins:
47                 print i,
```

```
48            exit()
49
50
51 ####  初始化 GPIO 口
52     def setup (self,pin) :
53         #采用 BCM 编号
54         GPIO.setmode (GPIO.BCM)
55         #设置 GPIO 为输出状态，输入低电平
56         GPIO.setup (pin,GPIO.OUT)
57         GPIO.output (pin,GPIO.LOW)
58
59     def loop (self,pin) :
60         for i in xrange (1,10) :
61             GPIO.output (pin,GPIO.HIGH)
62             print ("bell up")
63             time.sleep (self.up_time)
64             GPIO.output (pin,GPIO.LOW)
65             print ("bell down")
66             time.sleep (self.down_time)
67
68     def destroy (self,pin) :
69 ####  恢复 GPIO 口状态
70         GPIO.output (pin,GPIO.LOW)
71         GPIO.setup (pin,GPIO.IN)
72
73 ####  定义了 Light 类
74 class Light (object) :
75 ####  Light 类的构造函数
76     def __init__ (self,pin) :
77         self.pin = pin
78         self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]
79         self.up_time = 0.5
80         self.down_time = 0.5
81         self.check_pin (pin)
82         self.run()
83
84     def run (self) :
85         self.setup (self.pin)
86         try:
87             self.loop (self.pin)
88         except KeyboardInterrupt:
89             self.destroy (self.pin)
90
91     def check_pin (self,pin) :
92         if pin in self.pins:
93             print ("%d 是有效编号"%pin)
94         else:
95             print ("只能输入以下有效的 pin 编号")
96             for i in self.pins:
```

```
97              print i,
98          exit()
99
100
101 ####  初始化 GPIO 口
102     def setup(self,pin):
103 ####  采用 BCM 编号
104         GPIO.setmode(GPIO.BCM)
105 ####  设置 GPIO 为输出状态，输入低电平
106         GPIO.setup(pin,GPIO.OUT)
107         GPIO.output(pin,GPIO.LOW)
108
109     def loop(self,pin):
110         for i in xrange(1,10):
111             GPIO.output(pin,GPIO.HIGH)
112             print("light up")
113             time.sleep(self.up_time)
114             GPIO.output(pin,GPIO.LOW)
115             print("light down")
116             time.sleep(self.down_time)
117
118     def destroy(self,pin):
119 ####  恢复 GPIO 口状态
120         GPIO.output(pin,GPIO.LOW)
121         GPIO.setup(pin,GPIO.IN)
122
123 ####  定义 Ultrasonic 类
124 class Ultrasonic(object):
125     def __init__(self,trig_pin,echo_pin):
126         self.trig_pin = trig_pin
127         self.echo_pin = echo_pin
128         self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]
129         self.check_pin(self.trig_pin,self.echo_pin)
130         self.run(self.trig_pin,self.echo_pin)
131
132     def check_pin(self,trig_pin,echo_pin):
133         if trig_pin in self.pins:
134             print("%d 是有效编号"%trig_pin)
135         else:
136             print("只能输入以下有效的 pin 编号")
137             for i in self.pins:
138                 print i,
139             exit()
140         if echo_pin in self.pins:
141             print("%d 是有效编号"%echo_pin)
142         else:
143             print("只能输入以下有效的 pin 编号")
144             for i in self.pins:
145                 print i,
```

```
146            exit()
147
148
149 ####  初始化 GPIO 口
150     def setup(self,trig_pin,echo_pin):
151 ####  采用 BCM 编号
152         GPIO.setmode(GPIO.BCM)
153         GPIO.setwarnings(False)
154 ####  设置 GPIO 为输出状态，输入低电平
155         GPIO.setup(self.trig_pin,GPIO.OUT,initial=GPIO.LOW)
156         GPIO.setup(self.echo_pin,GPIO.IN)
157
158     def run(self,trig_pin,echo_pin):
159         self.setup(self.trig_pin,self.echo_pin)
160 ####  发出触发信号
161         GPIO.output(self.trig_pin,GPIO.HIGH)
162 ####  保持 15us
163         time.sleep(0.000015)
164         GPIO.output(self.trig_pin,GPIO.LOW)
165         while not GPIO.input(self.echo_pin):
166             pass
167 ####  echo 端口发现高电平，开始计时
168         t1 = time.time()
169         while GPIO.input(self.echo_pin):
170             pass
171 ####  echo 端口高电平停止，结束计时
172         t2 = time.time()
173         length = (t2-t1)*340/2
174         print("测试距离为 %0.2f m"%length)
175         return length
176
177 ####  定义 Alarm 类
178 class Alarm(object):
179     def __init__(self):
180         self.ptime = 5
181         self.tolerance = 0.05
182         self.mlog = mylog.MyLog()
183         self.run()
184
185 ####  通过超声波模块 2 次测距，如果测距的距离在可容忍的误差内则 Pass
186 ####  如果明显测距距离不一样，则说明门被打开，点亮报警灯，打开蜂鸣器报警
187     def run(self):
188         while True:
189             ul = Ultrasonic(23,24)
190             len1 = ul.run(23,24)
191             time.sleep(self.ptime)
192             len2 = ul.run(23,24)
193             if len1 > (len2 - 0.5) or len1 < (len2 + 0.5):
194                 self.mlog.info("初始位置 %f" %len1)
```

```
195                 self.mlog.info("目前位置 %f" %len2)
196                 self.echo()
197             else:
198                 pass
199
200     def echo(self):
201 ####  点亮报警灯
202         light = Light(12)
203 ####  打开报警器
204         bell = Bell(18)
205 ####  发送短信到手机
206         sendM = sendMsg.SendMail('警告','门已开')
207
208 if __name__ == '__main__':
209     al = Alarm()
```


注 意

这个 Python script 演示了两种模块调用的方法。这两种方法的效果都是一样的，喜欢哪种就用哪种。

好了，到了这一步已经差不多完成了。最后将这个自制的开门报警器放到距离门不超过 1.2m 的位置，执行命令：

```
cd
cd code/Python/alarm
sudo Python alarm.py
```

现在可以放心地关门离开了。Raspberry 会每 5s 检测一次与门之间的距离。如果门被打开则会立刻记入日志并发送短信息到手机上，如果觉得 5s 时间太长了，可以自行修改合适的间隔时间。如果有摄像头模块那就更好了，可以摄像存档。如果需要其他的功能，可以添加合适的模块，自行加入所需的功能。

第 8 章
实战：移动小车（手机控制+网页控制）

上章对单一模块进行了简单的组装应用，本章将对稍微复杂的模块进行组装应用，进一步了解 Raspberry 对模块的开发应用。本章组装一个 Raspberry 的移动小车，并让它运行。

本章主要内容包括：

- 移动小车所需要的硬件
- 移动小车的组装原理
- 实现移动小车的代码

8.1 硬件准备

移动小车，至少得包括一辆用电机驱动的小车。控制小车才轮得到 Raspberry 大显身手。所需的硬件大部分都在前面章节介绍过了，本章是对本书前面所讲的内容进行一次总结汇总。

8.1.1 必需的硬件

免驱无线网卡，如图 8-1 所示。



图 8-1 免驱无线网卡

移动小车（含直流减速电机 4 个），如图 8-2 所示。

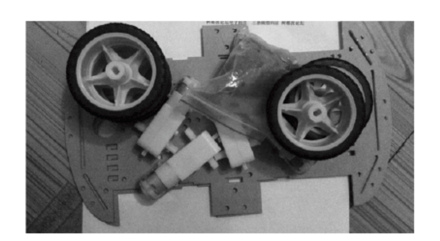


图 8-2　小车配件

L298N 电机驱动板模块，如图 8-3 所示。

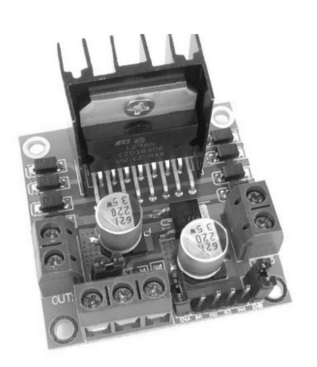


图 8-3　L298N 驱动模块

9V 电池一个，如图 8-4 所示。



图 8-4　电池

移动电源一个，如图 8-5 所示。



图 8-5　移动电源

8.1.2 可选的硬件

面包板，Raspberry pi2 扩展板。有了它们会轻松很多。如果没有也没关系。稍微麻烦点而已。超声波模块，有了这个可以添加移动小车的防撞功能。摄像头模块或者是摄像头，有了这个，可以实时地观看小车周边环境。

添加这些可选模块可以为小车增加更多的功能。

8.2 组装及原理

要想小车跑，就得先把小车组装好。本节详细地解说了移动小车的安装步骤和驱动原理。知其然后必知其所以然才能举一反三有所进步。

8.2.1 小车组装

1. 小车硬件

先将移动小车的包装拆开，清理配件。配件包括底盘 2 片，如图 8-6 所示。

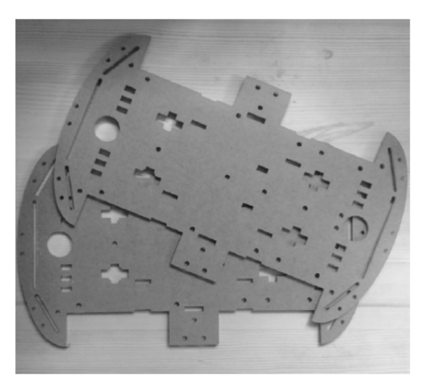


图 8-6 小车底盘

轮子 4 个，如图 8-7 所示。



图 8-7 小车车轮

测速码盘 4 个，如图 8-8 所示。

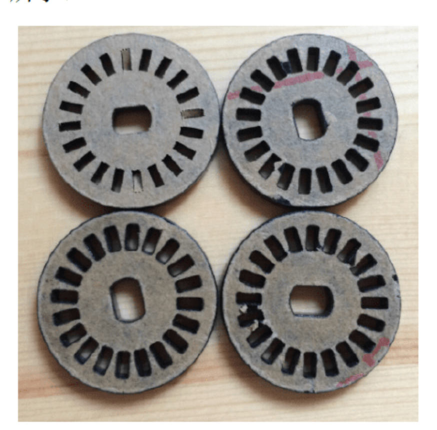


图 8-8　小车测速码盘

减速直流电机 4 个，如图 8-9 所示。



图 8-9　直流电机

M3x30 螺丝 8 个，如图 8-10 所示。



图 8-10　M3x30 螺丝

紧固片 8 片，如图 8-11 所示。

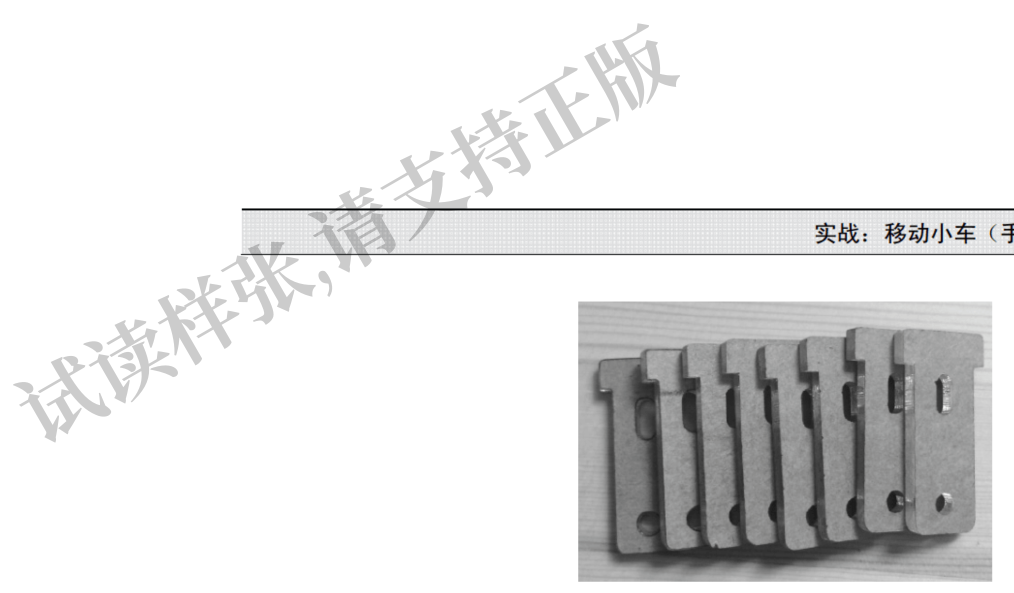


图 8-11　紧固片

M3x10 螺丝 6 个（有多的备件），如图 8-12 所示。



图 8-12　M3x10

M3 螺母 14 个（有多的备件），如图 8-13 所示。

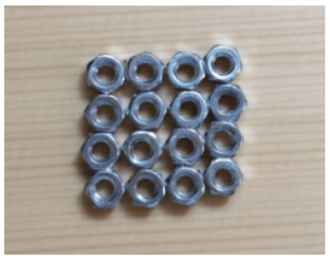


图 8-13　M3

铜柱 6 个，如图 8-14 所示。

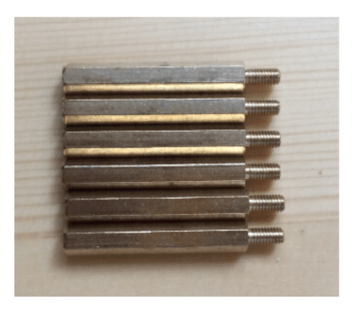


图 8-14　铜柱

2. 组装小车

先将底盘和固定片上的膜剥开。底盘和固定片本来应该是透明的，上面黄色的是一层贴膜。小心地将膜剥开，如图 8-15 所示。

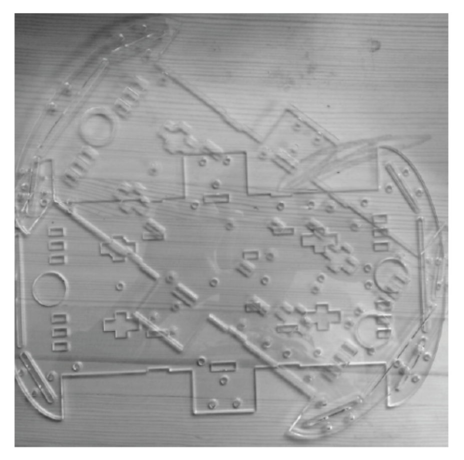


图 8-15　底盘（去包装）

剥开固定片的膜，如图 8-16 所示。

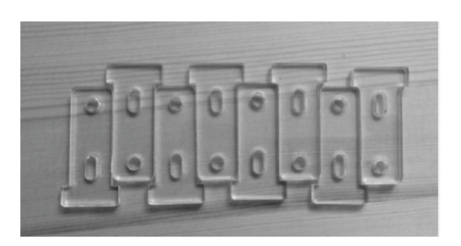


图 8-16　紧固片（去包装）

再用 2 个固定片将直流电机固定到底盘上，电机引线铜片朝外，即轮子的一端。如图 8-17 所示。

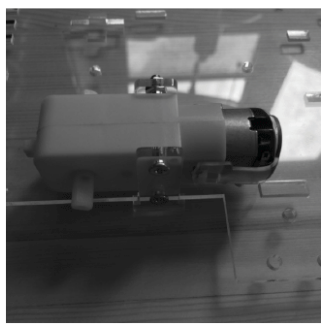


图 8-17　直流电机安装 1

按照以上的方法，依次将 4 个直流电机固定到底盘上，一定要将有铜片的那面朝外，即安装轮子的那面，如图 8-18 所示。



图 8-18　直流电机安装 2

将 4 个测速码盘安装到直流电机的内侧。如图 8-19 所示。

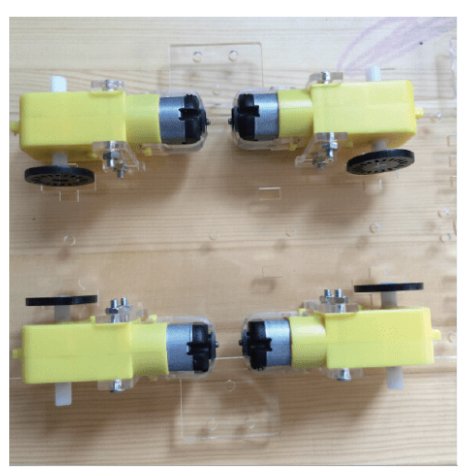


图 8-19　测速码盘安装

将 4 个轮子安装到直流电机上，如图 8-20 所示。



图 8-20　车轮安装 1

安装完 4 个轮子，将铜柱固定到小车的底板上，如图 8-21 所示。

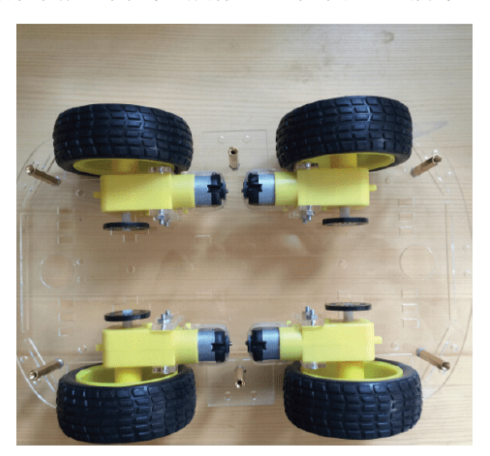


图 8-21　车轮安装 2

最后将顶盘用螺丝固定到铜柱上，如图 8-22 所示。

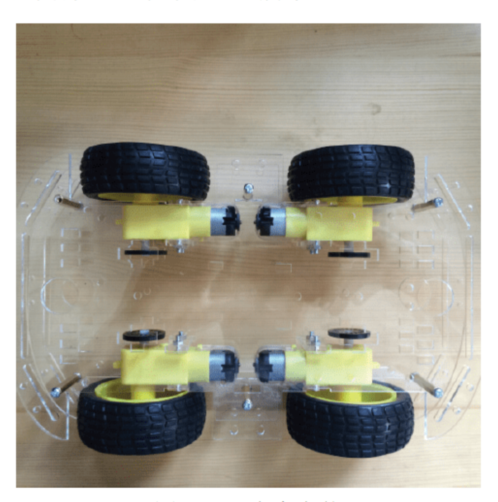


图 8-22　底盘安装

到这里小车已经基本安装完毕了，剩下的是与 L298N 电机驱动板模块的连接了。



注　意

小车的原理极其简单，就是 4 个直流电机控制 4 个轮子的前进后退。

8.2.2　电机组装

L298N 驱动模块很简单。它只有 4 个针脚控制直流电机的旋转，其他的几个针脚是使能端。

1. L298N 接线原理

L298N 的接入图，如图 8-23 所示。

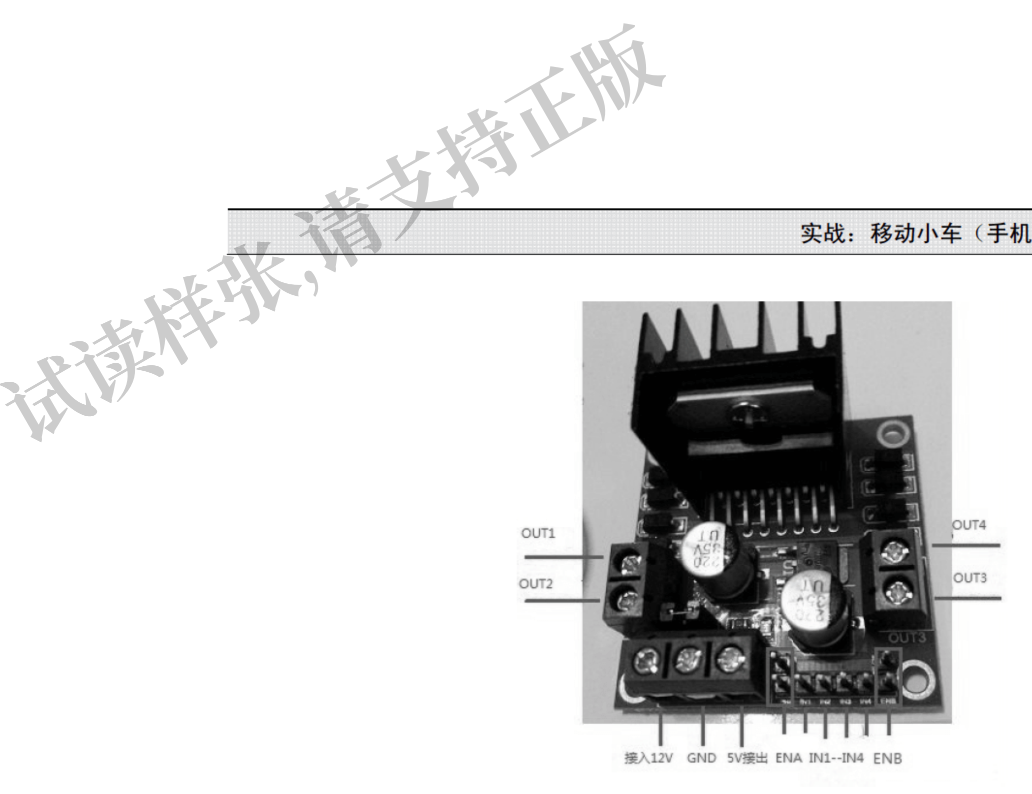


图 8-23　L298N 模块针脚

OUT1 和 OUT2 分别连接直流电机的正负极，同样 OUT3 和 OUT4 也是连接直流电机的正负极的。因为有 4 个直流电机，这里只打算用一个 L298N 电机控制，所以将同侧的两个直流电机并联到一起。这样的弊端就是同侧的电机会一起动作，没那么灵活。

接入 12V 就是直流电源的正极接入端，这里要求的是 12V，但实际上 7V-35V 都是没问题的。所以无须使用电池组了，直接给一个 9V 的电池就可以了。虽然动力不是很足，但小车本身不重，勉强可以带动。

GND 是接入直流电源的负极接入端，这就没什么好解释的了。

5V 接出是如果还有其他的设备需要供电，可以从这里取电供应，例如如果有个强劲的电池组则不需要移动电源给 Raspberry 供电，可以直接从这个端口给 Raspberry 供电。

ENA 和 ENB 分别为 A、B 电机的使能端，一开始 ENA 和 ENB 各自的上下两个针脚是用跳线帽连接起来的，拔掉就可以接线了。

IN1-IN4 分别是 A、B 电机的控制端。我们可以通过 Raspberry 上的 Python 脚本控制 GPIO 上的高低电平，间接地控制直流电机的转动与否。

2. L298N 连接直流电机

直流电机接线接口比较小，需要比较细的导线。可以将杜邦线稍微修改一下用于模块和电机之间部件的连接。

L298N 电机比较小巧，可以把它放到小车的两块面板之间，4 个直流电机的中间。打开小车的顶面板。先将直流电机连接到 OUT1、OUT2 接口，如图 8-24 所示。

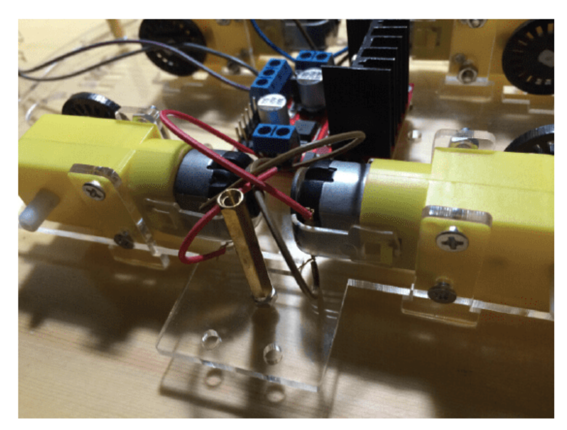


图 8-24　直流电机接线 1

同侧的两个直流电机的方向是相反的，接线的时候请注意方向。请看清楚杜邦线的颜色，最后接好的效果如图 8-25 所示。

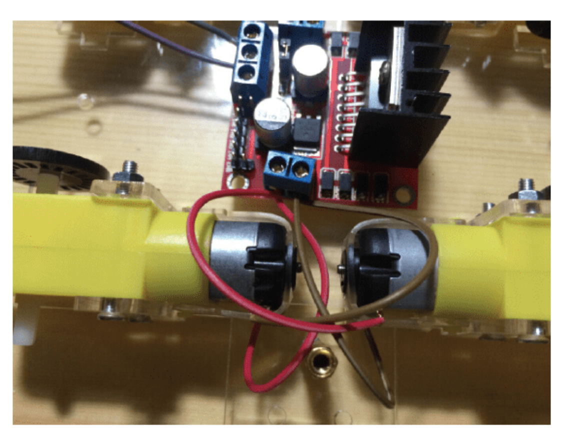


图 8-25　直流电机接线 2

将另一侧的两个电机按照顺序接好，最后电机的接入图，如图 8-26 所示。

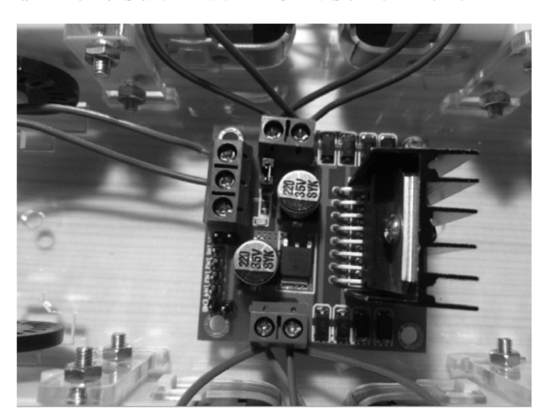


图 8-26　L298N 接线

3. L298N 连接 Raspberry/面包板

将 ENA、ENB 的跳线拔出，用杜邦线接入这 4 个端口，如图 8-27 所示。

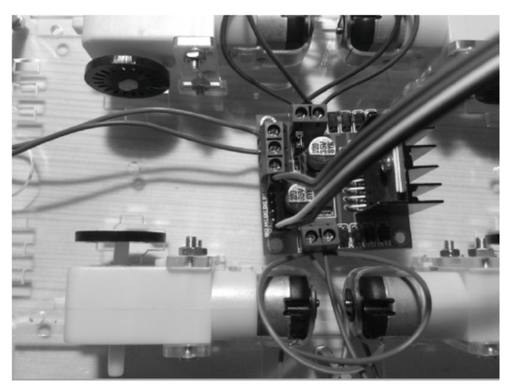


图 8-27　使能端接线

最后用杜邦线接入 IN1-IN4 这 4 个端口，如图 8-28 所示。

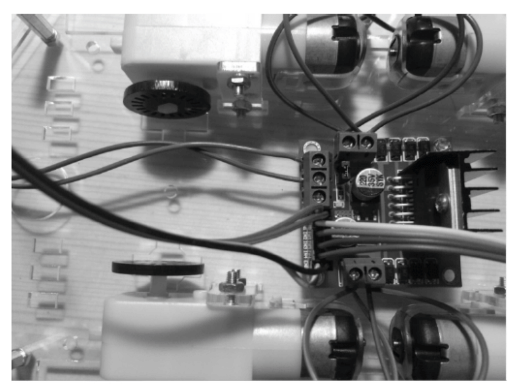


图 8-28　控制端接线

将所有接出的杜邦线理清顺序，从小车顶板的孔中穿出，将小车顶板固定。准备连接 Raspberry 设备，如图 8-29 所示。

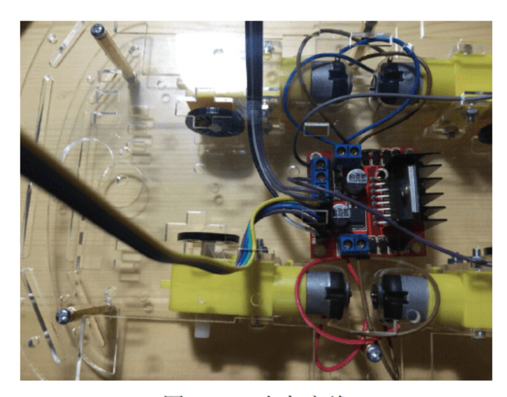


图 8-29　小车穿线

在这里使用了面包版、扩展板等辅助设备。如果没有这些请仔细核对 GPIO 的端口，避免不必要的麻烦。将连接 ENA 和 ENB 的 4 根杜邦线连接到了 5、6、13、19 号端口。将连接 IN1-IN4 的 4 个杜邦线连接到了 21、22、23、24 号端口，如图 8-30 所示。

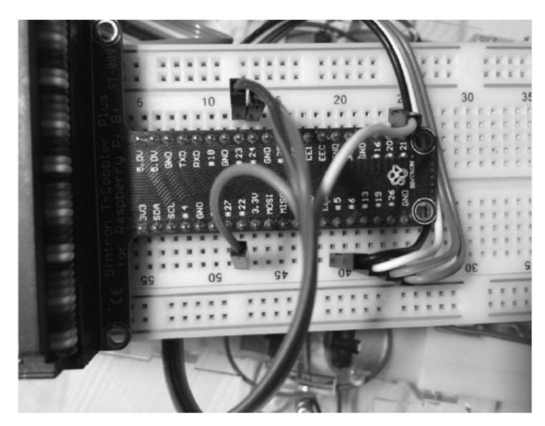


图 8-30 GPIO 接线



注 意

L298N 不设计到寄存器数据，只需要控制电平的高低，也是非常简单的。

4. L298N 电源输入

最后将连接电源接入口和 GND 的杜邦线连接到 9V 电池的正负极，如图 8-31 所示。

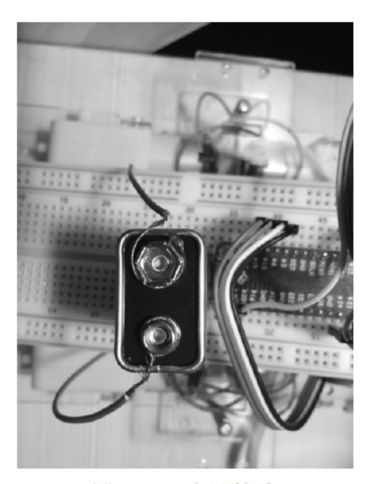


图 8-31 电池接线

将移动电源接入 Raspberry 的 miniUSB 接口，将无线网卡接入 Raspberry 的 USB 接口，小车硬件上已经准备完毕了，如图 8-32。

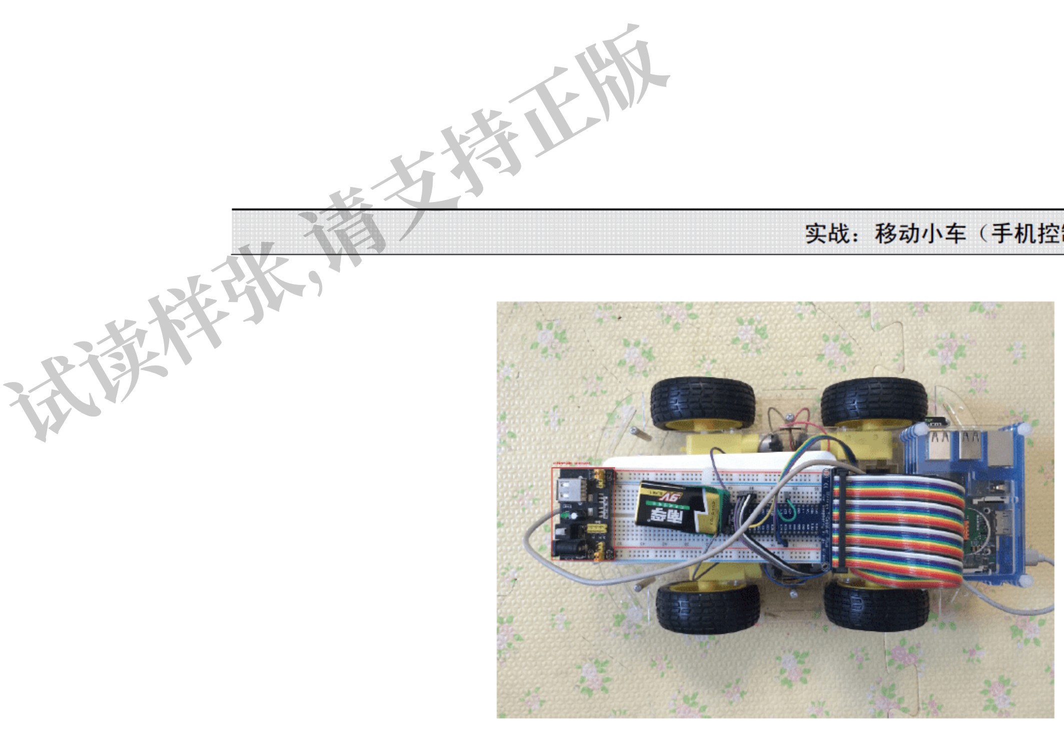


图 8-32　完整小车俯视图



注　意

可以直接从 Raspberry 上给 L298N 模块取电，省去 9V 的电池。这样的缺点就是小车运行起来慢得让人绝望。

8.2.3　小车原理

用 Raspberry 驱动 L298N 电机与上章的 Raspberry 驱动二极管、蜂鸣器、超声波模块可以说是没有任何区别。只是接入的线稍微多了一点点而已。

现在接入 Raspberry 的 GPIO 上的共有 8 根线。4 根线是使能端，这 4 个端口没什么好说的，直接给它个高电平就可以了，只有给它高电平直流电机才能运行。4 根线是连接 IN1-IN4 的，这才是操作 L298N 电机的关键。其中 IN1 给高电平，将控制 A 电机的前进。IN2 给高电平控制 A 电机的后退。IN3 和 IN4 控制的是另一侧 B 电机的前进后退。了解到了这点，控制小车就没有什么难度了。



注　意

还可以将超声波模块和摄像头加入进来。超声波模块做成防撞功能，视频模块可做成远程监控。

8.3　软件准备

硬件准备完毕，再来看软件方面。只有软硬结合才能驱动小车运行。本节将使用 Python 脚本控制小车的前进、后退、转弯，之后进一步使用 Web 页面来控制小车。

8.3.1　Python 控制

只需要控制 IN1~IN4 这几个端口的高低电平，就可以控制小车了。使用 Putty 登录 Raspberry，执行命令：

```
cd
```

```
mkdir -pv code/Python/car
cd $_
touch2py car.py
vi car.py
```

以下是 car.py 的代码：

```
 1 #!/usr/bin/env Python
 2 # -*- coding:utf-8 -*-
 3 #Author  :hstking
 4 #E-mail  :hstking@hotmail.com
 5 #Ctime   :2015/09/21
 6 #Mtime   :
 7 #Version :
 8
 9
10 import RPi.GPIO as GPIO
11 import time
12 import sys
13
14
15 ####  定义 Car 类
16 class Car（object）:
17    def __init__（self）:
18        self.enab_pin = [5,6,13,19]
19 ####  self.enab_pin 是使能端的 pin
20        self.inx_pin = [21,22,23,24]
21 ####  self.inx_pin 是控制端 in 的 pin
22        self.RightAhead_pin = self.inx_pin[0]
23        self.RightBack_pin = self.inx_pin[1]
24        self.LeftAhead_pin = self.inx_pin[2]
25        self.LeftBack pin = self.inx pin[3]
26 ####  分别是右轮前进，右轮退后，左轮前进，左轮退后的 pin
27        self.setup()
28
29 ####  setup 函数初始化端口
30    def setup（self）:
31        print "begin setup ena enb pin"
32        GPIO.setmode（GPIO.BCM）
33        GPIO.setwarnings（False）
34        for pin in self.enab_pin:
35            GPIO.setup（pin,GPIO.OUT）
36            GPIO.output（pin,GPIO.HIGH）
37 ####  初始化使能端 pin，设置成高电平
38        pin = None
39        for pin in self.inx_pin:
40            GPIO.setup（pin,GPIO.OUT）
41            GPIO.output（pin,GPIO.LOW）
42 ####  初始化控制端 pin，设置成低电平
43        print "setup ena enb pin over"
```

```
44
45 ####  fornt 函数，小车前进
46     def front (self) :
47         self.setup()
48         GPIO.output (self.RightAhead_pin,GPIO.HIGH)
49         GPIO.output (self.LeftAhead_pin,GPIO.HIGH)
50
51 ####  leftFront 函数，小车左拐弯
52     def leftFront (self) :
53         self.setup()
54         GPIO.output (self.RightAhead_pin,GPIO.HIGH)
55
56 ####  rightFront 函数，小车右拐弯
57     def rightFront (self) :
58         self.setup()
59         GPIO.output (self.LeftAhead_pin,GPIO.HIGH)
60
61 ####  rear 函数，小车后退
62     def rear (self) :
63         self.setup()
64         GPIO.output (self.RightBack_pin,GPIO.HIGH)
65         GPIO.output (self.LeftBack_pin,GPIO.HIGH)
66
67 ####  leftRear 函数，小车左退
68     def leftRear (self) :
69         self.setup()
70         GPIO.output (self.RightBack_pin,GPIO.HIGH)
71
72 ####  rightRear 函数，小车右退
73     def rightRear (self) :
74         self.setup()
75         GPIO.output (self.LeftBack_pin,GPIO.HIGH)
76
77 ####  定义 main 主函数
78 def main (status) :
79     car = Car()
80     if status == "front":
81         car.front()
82     elif status == "leftFront":
83         car.leftFront()
84     elif status == "rightFront":
85         car.rightFront()
86     elif status == "rear":
87         car.rear()
88     elif status == "leftRear":
89         car.leftRear()
90     elif status == "rightRear":
91         car.rightRear()
92     elif status == "stop":
```

```
93        car.setup()
94
95
96 if __name__ == '__main__':
97     main(sys.argv[1])
```

测试一下，执行命令：

```
Sudo Python ./car.py front
Sudo Python ./car.py leftFront
Sudo Python ./car.py rightFront
Sudo Python ./car.py rear
Sudo Python ./car.py rightRear
Sudo Python ./car.py leftRear
Sudo Python ./car.py stop
```

此时 4 个直流电机应该是运转正常的。

8.3.2 Web 控制和手机控制

如果光是使用 Putty 来控制小车，未免太麻烦了一点。现在用网页来控制小车，进而使用手机来控制小车才行。

使用 Putty 登录 Raspberry，执行命令：

```
Cd
cd www
mkdir car
cd $
ln -s /home/pi/code/Python/car/car.py
vi index.PHP
```

将上节的 car.py 链接到本地目录下，方便调用。以下是 index.PHP 的代码：

```
 1 <html>
 2 <head>
 3 <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
 4 <title> 小车控制</title>
 5 </head>
 6 <body>
 7
 8 <form action="" method="GET">
 9 <h2>小车控制</h2>
10 向前左转<input type="radio" name="radio" value="1">
11 前行<input type="radio" name="radio" value="2">
12 向前右转<input type="radio" name="radio" value="3">
13 <br/>
14 停车<input type="radio" name="radio" value="0" checked="checked">
15 <br/>
16 向后左转<input type="radio" name="radio" value="4">
17 后退<input type="radio" name="radio" value="5">
```

```
18 向后右转<input type="radio" name="radio" value="6">
19 <br/>
20 <input type="submit" name ="submit" value="OK">
21 </form>
22 <?PHP
23 $var=$_GET["radio"] ;
24 switch ($var)
25 {
26 case "0" :
27     exec("sudo Python ./car.py stop");
28     break;
29 case "1" :
30     exec("sudo Python ./car.py leftFront");
31     break;
32 case "2" :
33     exec("sudo Python ./car.py front");
34     break;
35 case "3" :
36     exec("sudo Python ./car.py rightFront");
37     break;
38 case "4" :
39     exec("sudo Python ./car.py leftRear");
40     break;
41 case "5" :
42     exec("sudo Python ./car.py rear");
43     break;
44 case "6" :
45     exec("sudo Python ./car.py rightRear");
46     break;
47 }
48 ?>
49 </body>
50 </html>
```

这里是使用 PHP 来调用 car.py，因为 car.py 需要用到/dev/mem，需要使用 root 权限，所以用网页调用 Python 脚本时也必须要有 root 权限。最简单的方法，把 www-data 加入/etc/sudoers 后面。执行命令：

```
sudo chmod 640 /etc/sudoers
sudo sed -i '$a www-data ALL=(ALL) NOPASSWD:\/usr\/bin\/Python' /etc/sudoers
sudo chmod 440 /etc/sudoers
```


注 意

在这里没有考虑到网络安全，仅作演示，实际操作中这种方法是极不可取的，正常情况应该是给 car.py 做个网络接口，这里只是采取最简单，最取巧的办法。

好了，现在已经将 www-data 加入到了 sudoers 文件中了。此后在网页中也可以调用 car.py 了。浏览器打开网页，如图 8-33 所示。

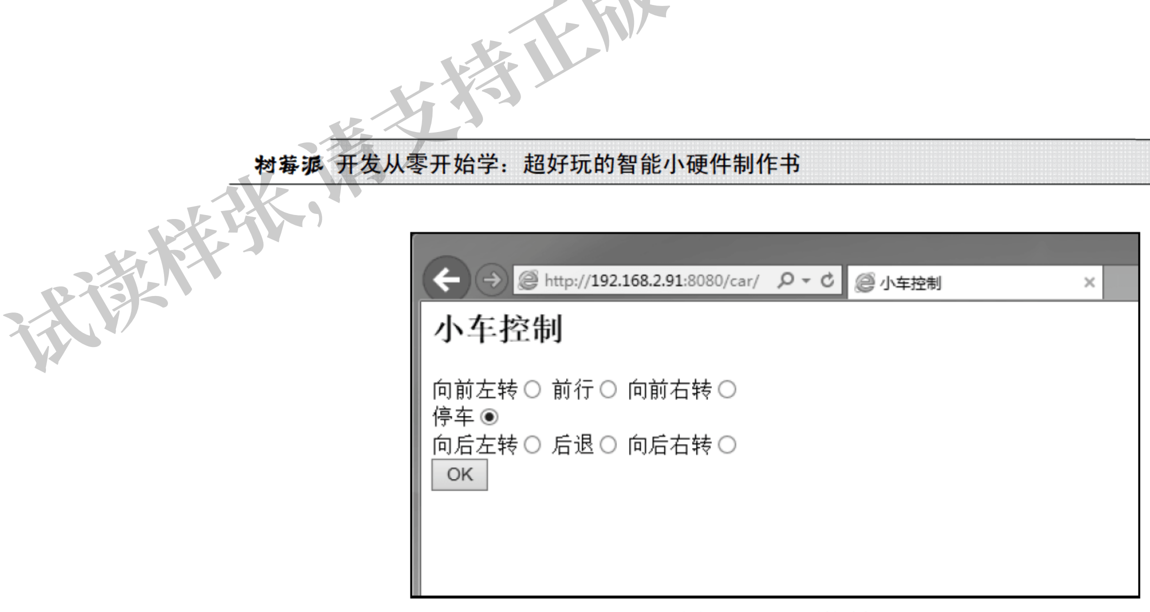


图 8-33　网页控制

测试一下，电机转动没问题，再来进行下一步。使用手机连接到本地的无线网络，使用 UC 浏览器打开网页（用哪个浏览器区别不大，chrome、safari 都是可以的），如图 8-34 所示。



图 8-34　手机控制

测试一下，现在手机也可以控制电机转动了。

8.3.3　无线设置

在上几节的测试中 Raspberry 都是连在有线网卡上测试的。既然叫车总不能拖着根线跑吧。下面我们用无线网卡来代替有线网卡。

使用 Putty 连接 Raspberry，执行命令：

```
sudo ifconfig -a
```

效果如图 8-35 所示。



图 8-35　wlan0

系统已经找到了无线网卡 wlan0，下面开始设置 wlan0，执行命令：

```
sudo vi /etc/network/interfaces
```

以下是最后修改好的 interfaces 代码：

```
auto lo
iface lo inet loopback

auto eth0
allow-hotplug eth0
#iface eth0 inet manual
iface eth0 inet static
address 192.168.2.91
netmask 255.255.255.0
gateway 192.168.2.1

auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
address 192.168.2.92
netmask 255.255.255.0
gateway 192.168.2.1
wpa-ssid yourssid
wpa-psk youpassword
```

重启系统，执行命令：

```
sudo reboot
```

好了，现在可以将有线网卡上的网线拔下来了。将小车的车轮安装好，电池充电器安装好，最后检查一遍，没有问题了。现在可以通过手机遥控小车了。